POLITECNICO DI TORINO

MASTER's Degree in INGEGNERIA MATEMATICA

MASTER's Degree Thesis

A CHANGE POINT MODEL FOR VARIABILITY ANALYSIS OF A MANUFACTURING PROCESS

Supervisors

Prof. MAURO GASPARINI

Prof. DANIELE CAPPELLETTI

Prof. ENRICO BIBBONA

Candidate

FRANCESCO PICASSO

MARCH 2021

Abstract

(ITA)

La tesi ha come obiettivo l'analisi di tempi ciclo provenienti da un macchinario per stampaggio a iniezione di materiale plastico al fine di catturare l'evoluzione temporale dei dati tramite un semplice processo stocastico, che possa essere utilizzato per fare previsioni sulla produzione.

I dati ottenuti presentano evidenti change point del valor medio, e un numero importante di valori che appaiono anomali. Una panoramica riguardante statistiche robuste ad outliers è stata effettuata per poi applicare un algoritmo per individuare in maniera robusta le locazioni dei change-point (a questo fine due classi di algoritmi sono state implementate e messe a confronto).

Il modello finale considera la variabilità indotta da ritardi di misurazione, stimandone il contributo sulla variabilità totale con un approccio prima frequentista, approssimato, e in seguito con un analisi più precisa di tipo bayesiano.

(ENG)

The main objective of the thesis is the analysis of plastic manufacturing cycle times coming from an injection moulding machine with the aim of developing a stochastic process which represents the evolution of the data over time and can be useful when it comes to predictions.

Since anomalies are found and the expected value of the cycle time may change throughout the data, the thesis presents an overview of robust statistics with the aim of properly developing a robust change-point model (for this purpose two classes of change-point algorithms are applied and compared).

The final model, where the impact of measurement delay on the process variability is taken into account, is built at first with a frequentist approach, based on a distribution approximation, then with a more precise bayesian approach.

Acknowledgements

I would like to express my very great appreciation to professor Mauro Gasparini, professor Daniele Cappelletti and professor Enrico Bibbona, my supervisors, for their infinite patience and their suggestions for this work.

I would like to thank the staff of Cluster Reply for the data given and for the opportunity to work on the thesis while learning how an office works.

I would like to offer my special thanks to my family, which supported me through these years, to my best friend, who's been by my side since high school and to all the friends that made these years in Turin unforgettable. The last acknowledgement is to my girlfriend, who supported me in a way that I would have never imagined.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The thesis is the analysis of a dataset retrieved, with the help of Cluster Reply srl, from an injection moulding machine from a plastic factory: the measurements are made using a sensor which detects the timestamps when the machine closes and opens, as well as the times when the machine turns on and off (the so called on-off cycles).

The dataset originally consisted in various JSON files, each one of them consists in a list of timestamps and signal changes.

The signal by which the cycle time is retrieved is "machine_open", meaning that a cycle time can be defined as the difference between two machine openings. When the machine turns on the first "machine_open" signal is interpreted as the beginning of the first cycle time, for the same reason the last cycle time measured finishes with the last "machine_open" signal before the machine turns off.

Another useful measure is the "machine_open_counter" signal which increments every time that "machine_open" goes to 1.

A processed version of the original data is presented and will be the base of all the data analysis.

```
N=nrow(pp.tot)
head(pp.tot)
```

```
##   i     time.n              time.s count
## 1 1 1590037860 2020-05-21 07:10:59 23190
## 2 1 1590037887 2020-05-21 07:11:26 23191
## 3 1 1590037914 2020-05-21 07:11:53 23192
## 4 1 1590037941 2020-05-21 07:12:20 23193
```

```
## 5 1 1590037967 2020-05-21 07:12:47 23194
## 6 1 1590037994 2020-05-21 07:13:14 23195
```

The dataset pp.tot consists of four variables:

- Variable "i" is the index of the on-off cycle of the event
- Variable "time.n" is a numeric conversion of the timestamp
- Variable "time.s" is the timestamp in format "AAAA-MM-DD HH:MM:SS,0S"
- Variable "count" is the updated number of the counter when an event occurs

This dataset is expected to be the realization of a counting process, so the first thing is to check if negative counter anomalies are present.

The following chunk of code will generate a matrix with as many rows as the counter anomalies that are found and three columns, useful to inspect the anomaly.

```
#check counter for errors/overflows
count.tot=pp.tot$count
#find anomalies
count.anom<-which(diff(count.tot)<0)+1

#inspect anomalies
t(sapply(count.anom,function(i) count.tot[(i-1):(i+1)]))
```

```
##       [,1] [,2]  [,3]
## [1,] 38955  112 55226
## [2,] 65535    0     1
## [3,]  6292  181  6427
```

It is easy to see that, out of the three anomalies revealed, the second is an overflow of a 16-bit counter (in fact $65535 = 2^{16} - 1$). This means that, in order to have a realization of a counting process, the number 2^{16} is added to every counter value after the overflow anomaly.

```
pp.tot[count.anom[2]:N,]$count<-pp.tot[count.anom[2]:N,]$count+2^16
```

The other anomalies have to be inspected a little better

```
count.tot=pp.tot$count
count.anom<-which(diff(count.tot)<0)+1
alply(count.anom,1,function(i) pp.tot[(i-1):(i+1),])[1:2]
```

```
## $`1`
##        i     time.n               time.s count
## 15759  9 1590601653 2020-05-27 19:47:32 38955
```

```
## 15760 10 1591356141 2020-06-05 13:22:20   112
## 15761 10 1591356150 2020-06-05 13:22:29 55226
##
## $`2`
##         i     time.n               time.s count
## 32255 72 1591793084 2020-06-10 14:44:44 71828
## 32256 73 1591796669 2020-06-10 15:44:28 65717
## 32257 73 1591796669 2020-06-10 15:44:29 71963
```

Both counter anomalies happen at the beginning of an on-off cycle, as testified by the changing of the variable "i", these events recorded are not likely to be the beginning of a cycle time so they will be simply removed from the data.

A plot of the manufacturing process is shown in order to check for further anomalies

```
pp.tot<-pp.tot[-count.anom,]
pprocess<-split(pp.tot,pp.tot$i)
pprocess<-lapply(pprocess,function(x) x[names(x) !="i"])
plot(pp.tot$time.n,pp.tot$count,xlab="Timestamp (numeric)"
,ylab="Counter",type="o")
```

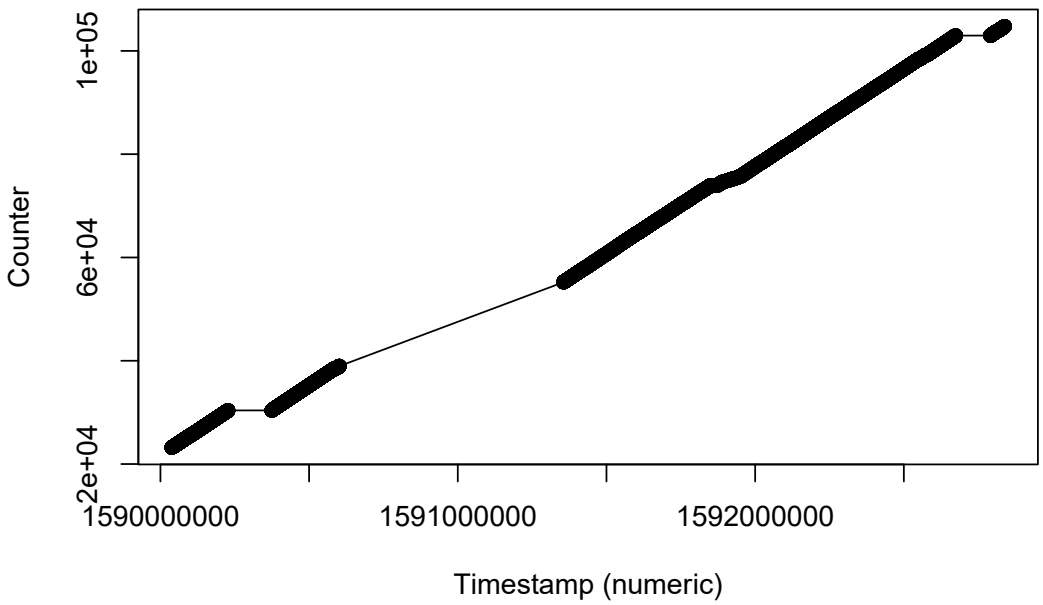

Figure 1.1: Counter-Timestamp plot

It can be seen that only positive increments are present, however positive anomalies have to be inspected yet. Usually positive anomalies are related to missing data in the manufacturing process, in fact the company noticed that the sensor had stopped working for some days.

```
count.tot=pp.tot$count
table(diff(count.tot))
```

```
##
##     1     2     3     5     6     7    14    23    51   135 16271
## 64962    24    11     3     1     1     1     1     1     1     1
```

The difference between counter values in almost the totality of cases is 1, though it could be seen, also from the previous results, that some data are missing. Next step is to find where these data are missing.

The most likely place to find missing data is between two on-off cycles,

```
count.miss<-which(diff(count.tot)>1)+1
list.miss<-lapply(count.miss,function(i) pp.tot[(i-1):(i+1),])
length(list.miss)
```

```
## [1] 45
```

The variable "count.miss" retrieves the position of the event following the missing data, while the list "list.miss" shows the rows of the dataset immediately before and after the missing data. It is not practical to inspect the list elements one by one, instead the change of the variable "i" concurrently with the missing data will be shown.

First of all there has to be a change in the variable "i" in each of these data chunks.

```
all(sapply(list.miss,function(l) length(unique(l$i)))==2)
```

```
## [1] TRUE
```

Once this is verified it can be seen that the missing data in each element of "list.miss" are between the first and the second row, an example is reported below

```
list.miss[[1]]
```

```
##      i     time.n              time.s count
## 7192 1 1590227162 2020-05-23 11:46:02 30381
## 7193 2 1590374338 2020-05-25 04:38:58 30388
## 7194 2 1590374364 2020-05-25 04:39:24 30389
```

What is expected, as in this case, is a change of the variable "i" from the first to the second row, meaning that the machine has been turned off, then on and this is

the cause of the missing data.

```
all(sapply(list.miss,function(l) diff(l$i)==c(1,0)))
```

```
## [1] TRUE
```

It can be now stated that all missing data are between on-off cycles. This statement means that a single on-off cycle in the manufacturing process is a counting process.

1.1 Inter-Event Times

The most interesting measures of this manufacturing process are the cycle times, which are the times between two "machine-open" events or inter-event times,

```
#list
iet.l<-dlply(pp.tot,"i"
,function(pp) data.frame(i=pp$i[-1]
                         ,iet=diff(pp$time.n)
                         ,t.start=pp$time.s[-nrow(pp)]
                         ,t.end=pp$time.s[-1]
                         ,count.end=pp$count[-1]))
#data frame
iet.tot<-do.call("rbind",iet.l)
n=nrow(iet.tot)
head(iet.tot)
```

```
##     i      iet              t.start               t.end count.end
## 1.1 1 27.41664 2020-05-21 07:10:59 2020-05-21 07:11:26     23191
## 1.2 1 26.70909 2020-05-21 07:11:26 2020-05-21 07:11:53     23192
## 1.3 1 26.85811 2020-05-21 07:11:53 2020-05-21 07:12:20     23193
## 1.4 1 26.68684 2020-05-21 07:12:20 2020-05-21 07:12:47     23194
## 1.5 1 26.80361 2020-05-21 07:12:47 2020-05-21 07:13:14     23195
## 1.6 1 26.64586 2020-05-21 07:13:14 2020-05-21 07:13:40     23196
```

The data frame "iet.tot" has 5 variables:

- Variable "i", met before, is the on-off cycle
- Variable "iet" is the inter-event time, which will be object of analysis
- Variables "t.start" and "t.end" are the timestamps of the events related to the inter-event time
- Variable "count.end" is nothing more than the counter value at "t.end"

1.2 Descriptive Statistics

Some descriptive statistics are made about the cycle time, in order to better understand its behaviour. The most trivial ones are mean and variance (expressed in seconds).

```
data.frame(mean.iet=mean(iet.tot$iet) ,var.iet=var(iet.tot$iet)
 ,max.iet=max(iet.tot$iet),min.iet=min(iet.tot$iet))
```

```
##   mean.iet  var.iet  max.iet   min.iet
## 1 26.60116 17.01496 138.3013 0.2648757
```

The variability of the process seems high, however it is very unlikely for a production time to be 100 times lower than its average, this suggests that some anomalies in this dataset can be present. In this case a boxplot can be useful for a preliminary anomaly detection.

```
boxplot(iet.tot$iet,horizontal = TRUE)
abline(v=mean(iet.tot$iet),col="red")
```

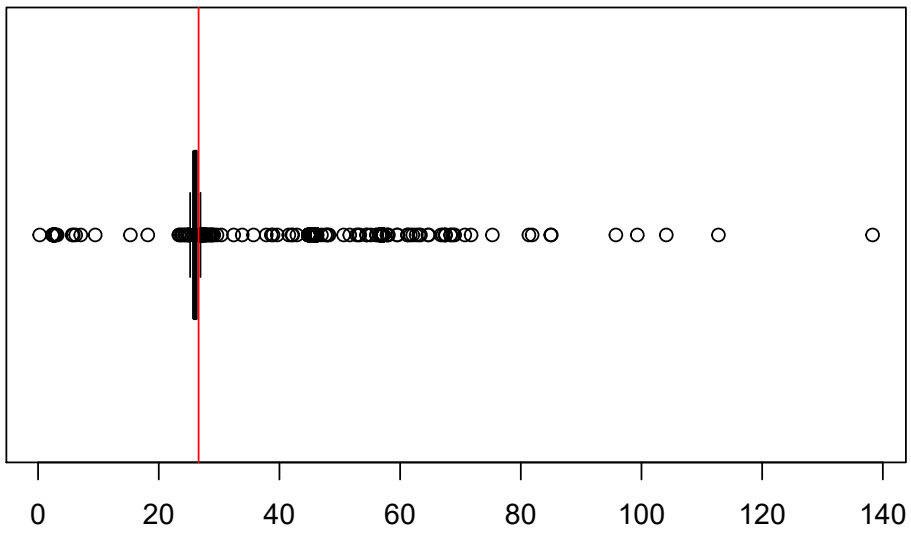

Figure 1.2: Cycle Time Boxplot

It can be seen that most of the data are narrowed near and slightly under the mean, while most of the variability comes from a decent amount of anomalies.

It can be of interest to capture the distribution of the majority of the cycle times in order to develop a model, for this purpose a kernel density plot of the central 90% of the data is computed.

```
plot(density(iet.tot$iet[iet.tot$iet<quantile(iet.tot$iet,0.95) &
 iet.tot$iet>quantile(iet.tot$iet,0.05)]),main="",
xlab="Inter-Event Time (s)")
```

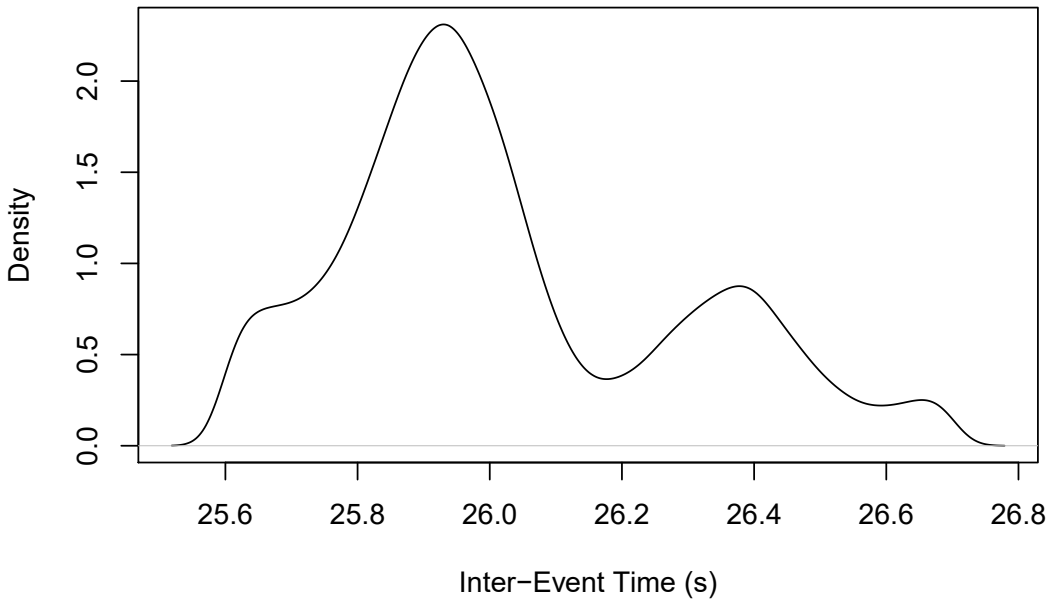

Figure 1.3: Density Plot of trimmed IET

The distribution is multimodal, this excludes some simple models for a manufacturing cycle time, like a nice distribution fitting.

Another tool which can be helpful in choosing a model for the cycle times can be the (robust) autocorrelation plot, which gives some information since most of the inter-event times are consecutive. The function `acfrob`, from package `robts` [1][2] will be further discussed in Section 2.4

```
acfrob(iet.tot$iet,approach="rank")
```

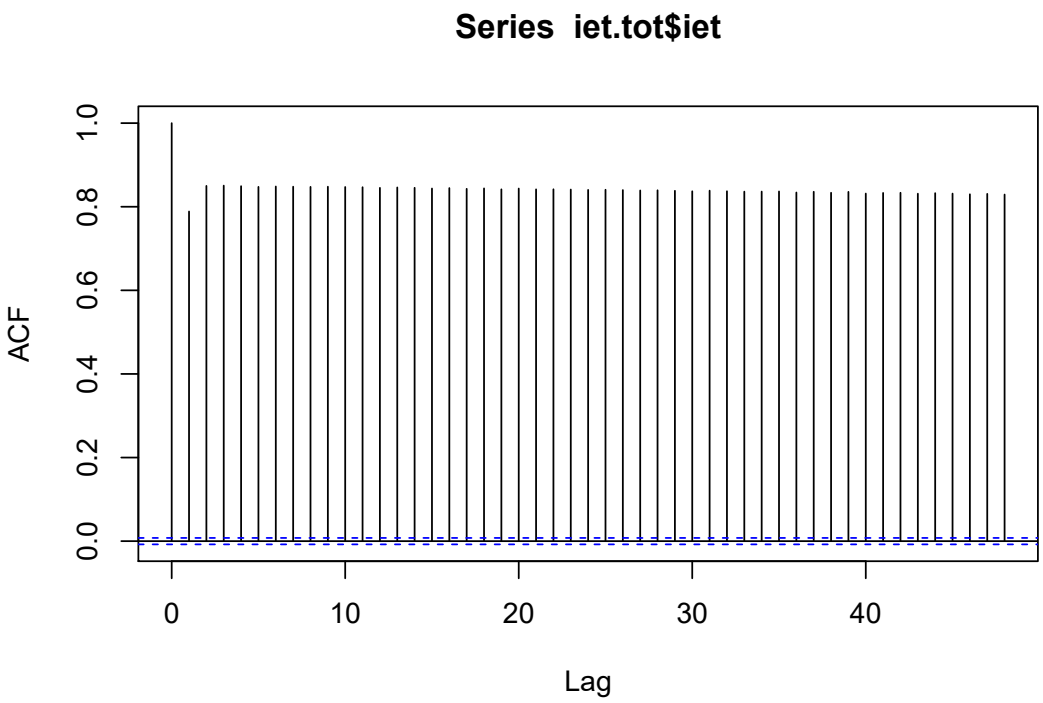

Figure 1.4: Spearman's ACF of the Cycle Times

The plot represents a strong correlation between variables for all lags considered, which can be the result of non-stationary cycle times. This means that more detailed analysis of the variables over time should be done.

1.3 Cycle Times over time

Given the results of the auto-correlation plot, it is clear that most descriptive statistics are not helpful in finding a sensible fit for the data.

The variable "machine_open_counter" plays now a more important role, since it tells the evolution of the cycle times over time. In order to better understand the inter-event time behaviour in this process a plot is shown with the counter on the x axis and the IET on the y axis

```
plot(iet.tot$count.end,iet.tot$iet,xlab="Counter (end)",ylab="Cycle Time")
```

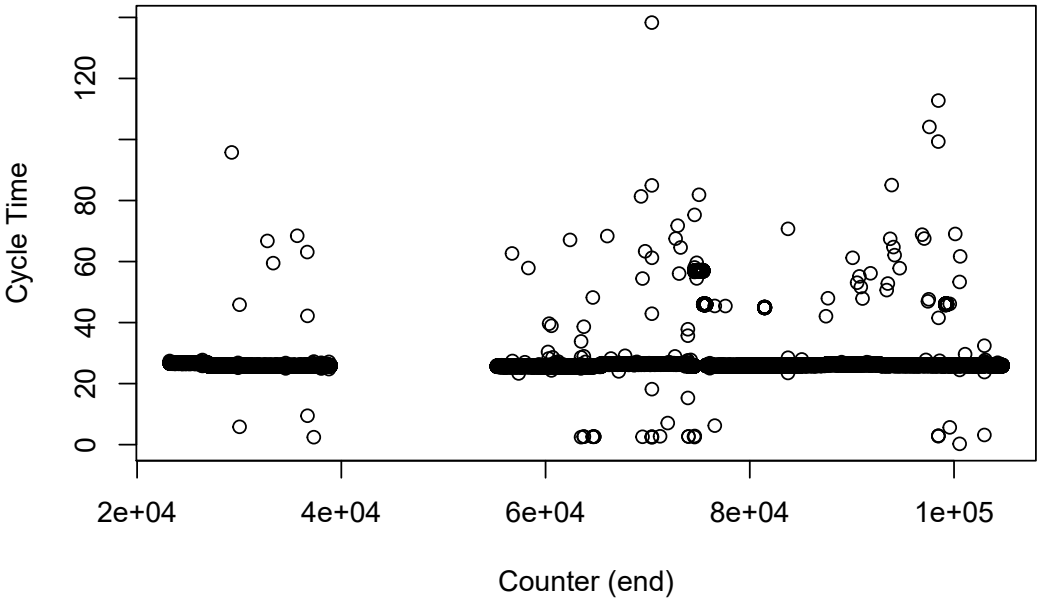

Figure 1.5: Counter - Cycle Time plot

By seeing this plot the leap caused by missing data is evident, what is also evident is the presence of a wide range of values for the cycle time, this usually denotes the presenct of outliers.

Another plot is shown where missing data are not considered.

```
plot(iet.tot$iet,type="l",ylab="Cycle Time")
```

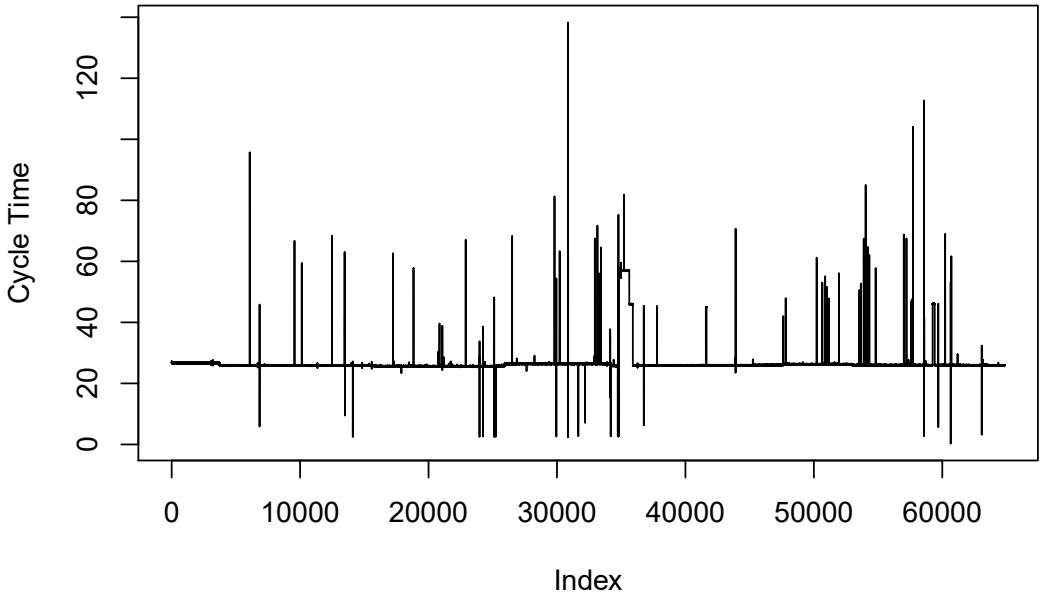

Figure 1.6: Evolution of Cycle Time, not counting missing data

By seeing this plot the first interesting thing about inter-event times comes out: there are changes in the mean of the cycle time, this is particularly evident between index 30000 and 40000.

Now, in order to further inspect the data a plot of the cycle times between 30000 and 40000 is shown, where the vertical red lines represent any cycle time after which the machine is turned off.

```
plot(30000:40000,iet.tot$iet[30000:40000],type="l",xlab="Index"
,ylab="Cycle Time")abline(v=which(diff(iet.tot$i)==1)+1,col="red")
```

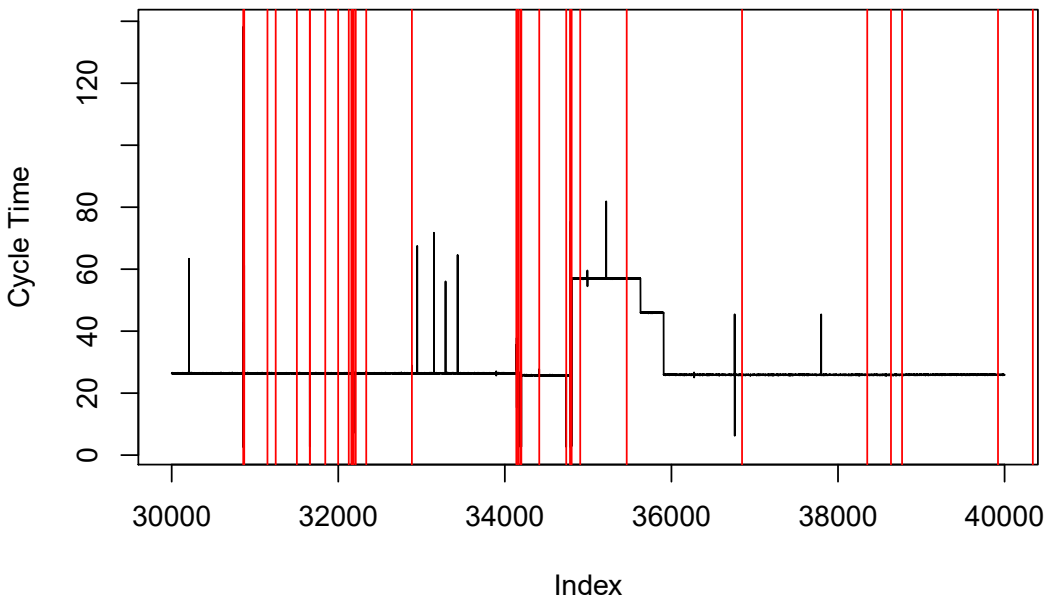

Figure 1.7: Zoomed plot with on-off cycles

This plot tells not only that some changes in mean are present, they also happen to be in the middle of the manufacturing process, not only when the machine is turned off then on.

The main results of a preliminary analysis are:

- Some structural breaks in the mean processing time are present and their positions are not known. A change-point model can explain a good part of the cycle time variability.
- Some outliers are present in the dataset, this means that most of the traditional statistical models do not work and there is the need of a robust model in order to fully explain the processing time variability.

1.4 Preprocessing

It is clear from the results listed that a change-point model has to be used even for preprocessing.

The first assumption, which is reasonable when dealing with a manufacturing process, is that the inter event times are independent. The second assumption is that the process variance, not considering outliers doesn't change over time.

Having this in mind the standard deviation can be estimated with a quantile-based estimator such as MAD, which will be discussed in Chapter 2

```
s<-iet.tot$iet
sigmah<-mad(diff(s)/sqrt(2))
sigmah
```

```
## [1] 0.1110669
```

In order to prove the fact that a robust change-point model has to be implemented , the first model will be based on the simple quadratic (L2) loss and a BIC-like penalty. Being this model sensitive to outliers, the expected result of the fitting will consist in more segments than expected, some of which are composed of only one point (which is almost certainly an outlier).

The non-robust model is applied and the number of change-point estimated will be shown. The function `rob_seg.std` from package `robseg` [3] will be discussed further in Section 4.2

```
s_<-s/sigmah
cpm.l2<-Rob_seg.std(s_,loss="L2"
                            ,lambda=3*log(n))
cpm.l2$K
```

```
## [1] 261
```

```
plot(s,ylim=c(20,60))
lines(cpm.l2$smt*sigmah,col="red")
abline(v=cpm.l2$t.est[-cpm.l2$K],col="blue")
```

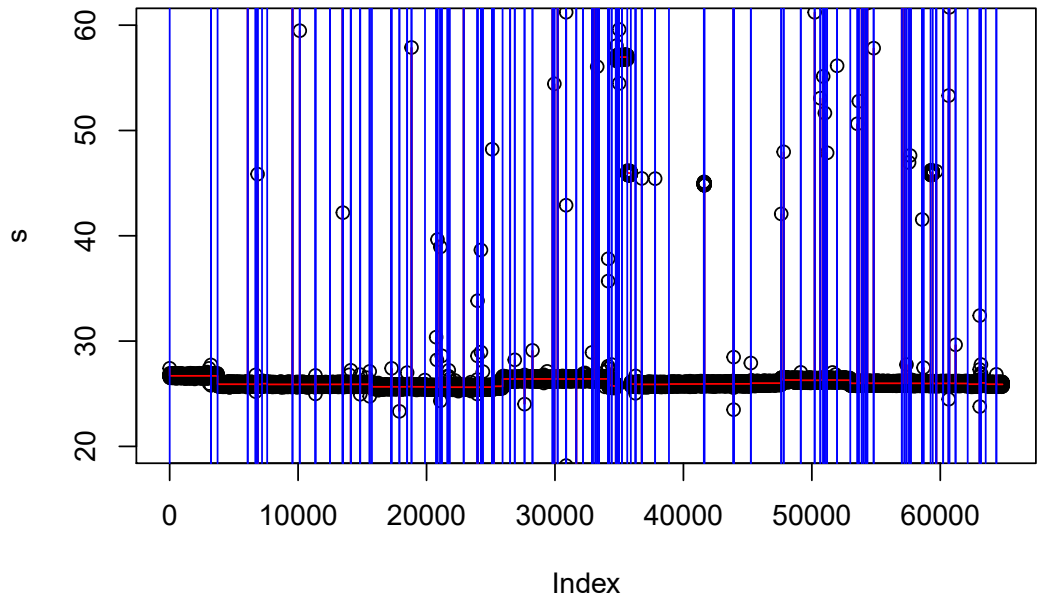

Figure 1.8: Change-points in L2 model

It is clear that this model doesn't fit well, since most change-points are actually caused by the outliers in the data.

The segments caused by outliers have generally length equal to one, it can be seen how many of them have been found.

```
sum(diff(c(0,cpm.l2$t.est))==1)
```

```
## [1] 142
```

The necessity of a robust model is proven by the inadequacy of the L2 model, for this purpose a model which uses a robust statistic is applied now for preprocessing and discussed later in Chapter 4

```
cpm.biw.prep<-Rob_seg.std(s_,loss="Outlier"
            ,lambda=3*log(n) ,lthreshold=3)
```

```
data.frame(n.cp=cpm.biw.prep$K,min.dist=min(diff(cpm.biw.prep$t.est)))
```

```
##   n.cp min.dist
## 1   28       58
```

The minimum segment length is sufficiently high to tell that groups of outliers are not classified as segments.

In order to verify the resistance to outliers of this preprocessing model a plot of the data is shown.

```
plot(s,ylim=c(20,60))
lines(cpm.biw.prep$smt*sigmah,col="red")
abline(v=cpm.biw.prep$t.est[-cpm.biw.prep$K],col="blue")
```

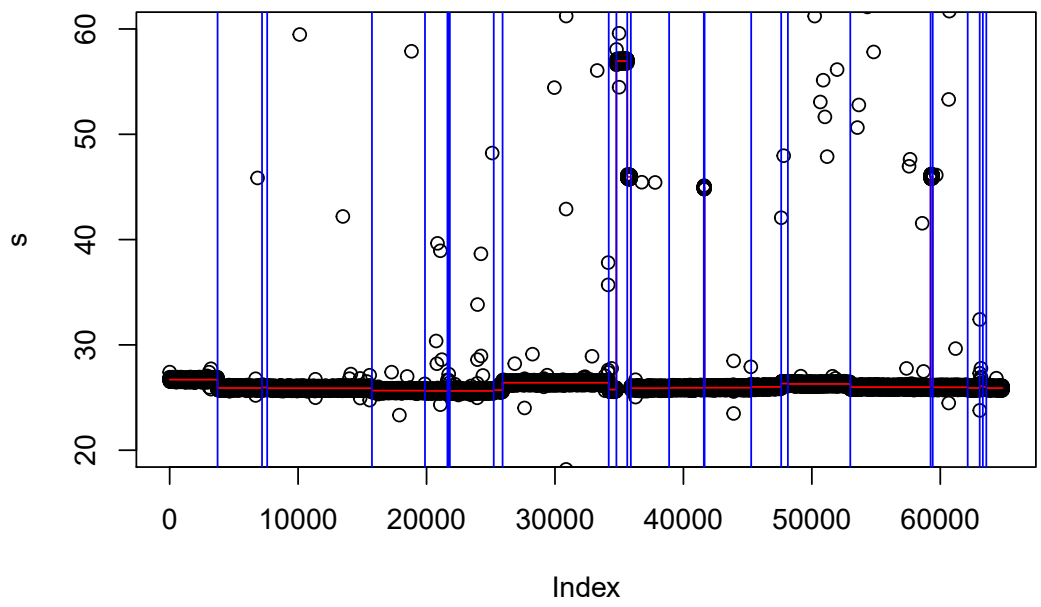

Figure 1.9: Change-Points in Robust Model

By looking at the plot, where the blue vertical lines represent change-point locations, the proposed preprocessing model can be actually useful in order to detect outliers and to get more information about the distributions of the inliers.

Once the change-point model is done, the means for each segment are found and the centered data (per segment mean) are obtained and plotted

```
smt<-cpm.biw.prep$smt*sigmah
res<-(s-smt)

plot(res,ylim=c(-3*sigmah,3*sigmah))
```

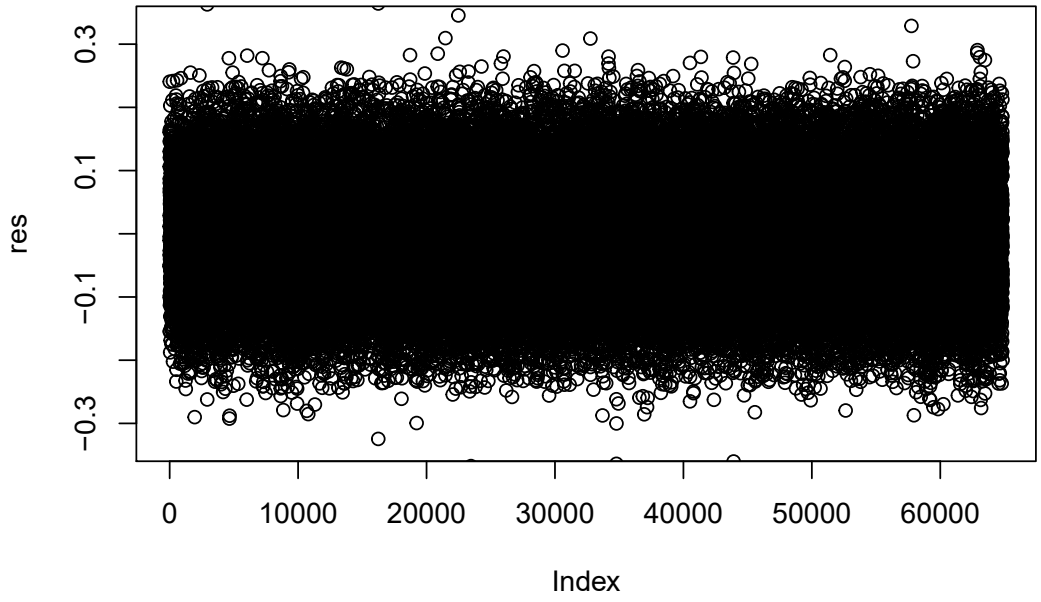

Figure 1.10: Centered Data Plot

In an interval of $\pm 3\sigma$ there do not seem to be any abrupt changes in variance. A model with inlier variance independent by the segments can be taken into consideration.

Another hypothesis that has to be verified for the centered data is independence between the cycle times, which requires no autocorrelation. For the purpose a robust correlogram of the centered data is made. The method used for the autocorrelation is Spearman's rho, which will be discussed later in Section 2.4, along with the GK estimator, which, in this case, gives similar results.

```
acfrob(res,approach="rank",cor.method="spearman")
```

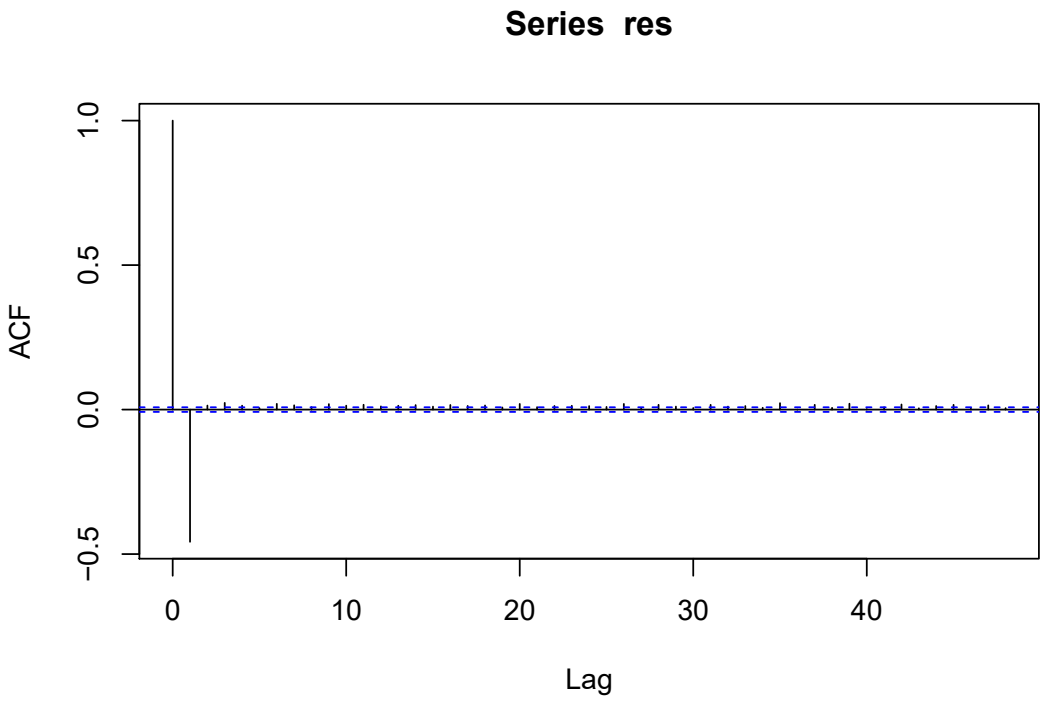

Figure 1.11: Autocorrelation function of centered data

Figure 1.11 shows a significant autocorrelation for lag 1, which breaks the independence hypothesis. The consequences of this are:

1. The model of the manufacturing process is not adequate as it assumes independence.

2. Due to the negative autocorrelation at lag 1, the estimator of the standard deviation based on differences $\hat{\sigma} = MAD\left(\frac{\Delta^-}{\sqrt{2}}\right)$ overestimates the scale parameter.

3. Most Change-point models assume independence between the data in a segment, this means that some attention has to be payed when fitting a change-point model.

A more sensible estimate of the standard deviation can be made by recalling the previous model and using a robust estimator of the standard deviation on the variable `res`

```
print(sigmah<-scaleTau2(res))
```

```
## [1] 0.08965183
```

1.5 Outlier Analysis

```
th=4.5*sigmah
iet.tot<-cbind(iet.tot,outl=abs(s-smt)>th)
outl<-which(abs(s-smt)>th)
plot(outl,s[outl],xlab="# items produced"
     , ylab="Inter-Event Time")
lines(smt,col="red")
```

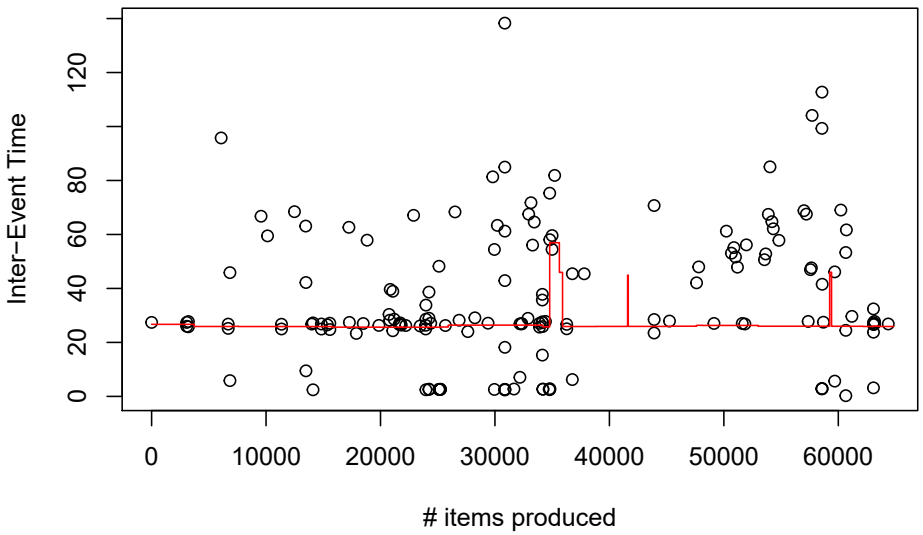

Figure 1.12: Outliers Plot

From this plot it is clear that outliers are almost homogeneous in the data (despite the segments around 40k items are much more regular than the rest).

Now a data frame summarizing any outlier in the dataset will be created.

Variable "pos" is the number of the items produced, which is also the position of the outlier in the dataset, "value" represents the actual value of the outliers,"mean" is the mean of the segment in which the outlier is found, "n.seg" is the number of such segment and dev is the deviation of the outlier from its mean.

```
outliers<- data.frame(pos=outl,mean=smt[outl])
outliers<-cbind(iet.tot[outl,],outliers,dev=(s-smt)[outl]
                ,n.seg=sapply(outl,function(x)
                                  sum(x>cpm.biw.prep$t.est))+1)
head(outliers)
```

```
##      i      iet             t.start               t.end count.end
## 1    1 27.41664 2020-05-21 07:10:59 2020-05-21 07:11:26     23191
## 3068 1 27.40986 2020-05-22 05:55:40 2020-05-22 05:56:07     26258
## 3069 1 25.97131 2020-05-22 05:56:07 2020-05-22 05:56:33     26259
## 3214 1 27.73439 2020-05-22 07:00:39 2020-05-22 07:01:07     26404
## 3215 1 25.79535 2020-05-22 07:01:07 2020-05-22 07:01:33     26405
## 6077 1 95.75740 2020-05-23 03:43:33 2020-05-23 03:45:08     29267
##        pos      mean        dev n.seg
## 1        1  26.69729  0.7193524     1
## 3068  3068  26.69729  0.7125730     1
## 3069  3069  26.69729 -0.7259829     1
## 3214  3214  26.69729  1.0370981     1
## 3215  3215  26.69729 -0.9019459     1
## 6077  6077  25.90339 69.8540147     2
```

Some peculiar patterns can be noticed by looking at this dataframe:

- There are groups of consecutive outliers.
- Some of the groups are composed by a high outlier followed by a low outlier for which the sum of the deviations from the mean is around 0.

1.5.1 Consecutive Outlier Analysis

First it has to be known how many groups of consecutive outliers are in the dataset. In order to detect them a list of these groups is created

```
out.cons<-list()
temp<-NULL
```

```
for(i in 1:(length(outl)-1)){
  temp<-rbind(temp,outliers[i,])
  if(diff(outliers$count.end)[i]!=1)
  {
    temp<-cbind(temp,cumdev=cumsum(temp$dev))
    if(nrow(temp)!=1) out.cons[[length(out.cons)+1]]<-temp

    temp<-NULL
  } }
rows.out<-sapply(out.cons,nrow)
data.frame(n.cons=length(out.cons)
,perc.cons= sum(rows.out)/length(outl)*100
           ,min.group=min(rows.out),max.group=max(rows.out))
```

```
##   n.cons perc.cons min.group max.group
## 1     31  40.96386         2         4
```

The list has 31 distinct groups and covers more than 40% of all outliers. Among them it is interesting to find the groups of consecutive outliers $\boldsymbol{W}$ that do not reject the hypothesis

$$\sum_{i}^{M} W_i - M\mu = 0$$

where M is the cardinality of a single group and Δ_i is a non-outlier inter-event time. Roughly speaking: the deviations within a single group of outliers have to almost cancel each other. Another condition has to be met: the cumulative deviation cannot be significantly lower than zero.

A possible explanation can be a gross error in a single measure while the manufacturing process is still going under control, as if it was an outlier for the measurement delay ε_i. A check must be done in order to verify whether a piece has been actually produced. The other outliers may be related to the actual manufacturing process more than to the measurement system.

```
#estimates of the standard deviation of the sum of i consec residuals
#CAREFUL, BD point is not 50%
sh=sapply(1:max(rows.out)
          ,function(i) {
            w=numeric(n-i)
            for(j in 1:(n-i)){
              w[j]=sum(res[j:(j+i)])
            }
```

```
            mad(w)
          } )

alpha=0.05

df.consdevs<-data.frame(
  mindev=sapply(out.cons,function(x) min(abs(x$cumdev)))
  ,mindev.h0=sapply(out.cons,function(x)
    min(abs(x$cumdev))<qnorm(1-alpha/2)*sh[which.min(abs(x$cumdev))])
  ,pos.min=sapply(out.cons,function(x) which.min(abs(x$cumdev)))
  ,n.in.group=rows.out
  ,prev.dev.pos=sapply(out.cons,function(x) {
  pos=which.min(abs(x$cumdev))
  ifelse(pos==1,return(F),return(all(x$cumdev[1:(pos-1)]>0)))
}))
head(df.consdevs)
```

```
##       mindev mindev.h0 pos.min n.in.group prev.dev.pos
## 1 0.01340992      TRUE       2          2         TRUE
## 2 0.13515227      TRUE       2          2         TRUE
## 3 0.17758828      TRUE       2          2         TRUE
## 4 0.11218375      TRUE       2          2         TRUE
## 5 0.05138215      TRUE       2          2         TRUE
## 6 0.10738906      TRUE       2          2         TRUE
```

Now the groups of measurement outliers are isolated, the next code chunk shows how many of these groups are present, how many outliers have been detected and their percentage over all data.

```
symm<-which(df.consdevs$mindev.h0 & df.consdevs$prev.dev.pos)

list.symm<-lapply(symm
       ,function(i) out.cons[[i]][1:(df.consdevs$pos[i]),] )
n.out.meas<-sum(sapply(list.symm,nrow)-1)
data.frame(n.groups.meas=length(symm),
n.out.meas=n.out.meas,perc.out.meas=n.out.meas/N*100)
```

```
##   n.groups.meas n.out.meas perc.out.meas
## 1            15         16     0.0246116
```

It can happen that, in a group of consecutive outliers, some are symmetric and some are not. The variable "remaining" collects any non-symmetric outlier in a group where symmetric ones are present.

```
remaining <- lapply(symm
      ,function(i)
       ifelse(df.consdevs$pos[i]<df.consdevs$n.in.group[i],
       out.cons[[i]][(df.consdevs$pos[i]+1):(df.consdevs$n.in.group[i]),]
       ,0))
remaining<-remaining[-which(sapply(remaining,function(x) x==0))]
remaining
```

```
## list()
```

In this case there are no outliers that satisfy this condition.

Now a list containing any non-symmetric outlier is created.

```
out.nsc.l<-merge.lists(out.cons[-symm]
,remaining[which(sapply(remaining,nrow)>1)])
out.notcons<-outliers[-unique(c(which(diff(outliers$count.end)==1)
                                ,which(diff(outliers$count.end)==1)+1)),]
out.notcons<-cbind(out.notcons,cumdev=out.notcons$dev)
out.list<-merge.lists(out.nsc.l,as.list(split(out.notcons,
seq(nrow(out.notcons)))))
```

1.5.2 On-off cycle edge outliers

Now the focus is on those (groups of) outliers that appear right after the machine turns on or right before it turns off. The causes of these anomalies have to be taken into consideration, however, when it comes to study the in-control process, they can be discarded since the time-series of the production can be started later or ended earlier.

```
edge.end<-!sapply(out.list,function(df)
(df$count.end[nrow(df)]+1) %in% iet.tot$count.end)

edge.start<-!sapply(out.list,function(df)
 (df$count.end[1]-1) %in% iet.tot$count.end)

outl.left<-do.call("rbind",out.list[!(edge.end | edge.start)])
```

This kind of outliers is discarded from the vector `pprocess`.

```
ts.out.start = do.call("rbind",out.list[edge.start])$t.start
ts.out.end = do.call("rbind",out.list[edge.end])$t.end

pprocess<-lapply(pprocess,function(df)
df[!(df$time.s %in% c(ts.out.end,ts.out.start)),])
```

```
pprocess<-pprocess[sapply(pprocess,nrow)>1]
pprocess<-lapply(1:length(pprocess),
function(n) cbind(i=n,pprocess[[n]]))
pp.tot<-do.call("rbind",pprocess)
```

So the inter-event times can be updated.

```
iet.tot<-iet.tot[-do.call("c",sapply(out.list[(edge.end| edge.start)],
function(x) x$pos)),]
s<-iet.tot$iet
```

The result is a preprocessed dataset where this type of outliers is removed. Mind that these outliers can be important in a more accurate outlier analysis, hence in that case they cannot be discarded. The analysis will be aimed on finding the best robust segmentation for the preprocessed data and, for this purpose, some robust statistics and robust change-point models are introduced in the next chapters.

Chapter 2

Robust Statistics

Real data often contain a main group of points following some distribution F and some observations that differ very significantly from the main group, following a distribution Δ, generally with very large variability: these observations are called outliers. Generally the former distribution is the most interesting one and, in order to estimate its parameters, some attention has to be payed to outliers since their presence can drastically affect the results of the most common estimators. In order to solve this issue some robust estimators are presented in this chapter, along with some description and measures of efficiency and robustness.

2.1 Breakdown Point as a measure of robustness

In order to find the most suitable estimator given a data set, a quantitative measure of robustness and resistance to outliers is needed. A simple and useful measure of robustness can be the minimum percentage of outliers needed to ruin the performance of an estimator, this is the key concept of the breakdown point, first introduced by Hampel in 1968 [4].

Definition 2.1.1. *Consider a sample $\boldsymbol{X}$ with j elements coming from an "outlier" distribution $\Delta(x)$ and $n-j$ elements coming from a distribution $F(x)$, let $\hat{\theta}(\boldsymbol{X})$ be an estimator of a parameter θ, with parameter space $\boldsymbol{\Theta}$. The finite-sample breakdown point is defined as the maximum ϵ^* so that*

$$\hat{\theta}(\boldsymbol{X}) \cap \partial\boldsymbol{\Theta} = \emptyset \quad \forall \Delta(x), \forall \frac{j}{n} < \epsilon^*$$

This quantity is generally a function of the sample size n: for example the finite-sample Breakdown Point of the sample mean estimator is $\frac{1}{n}$, meaning that a single

observation over n in the sample is sufficient to "ruin" the estimate.

2.1.1 Asymptotic Breakdown Point

Even more interesting is the asymptotic behavior of the breakdown point. By choosing $n \to \infty$ it can be proven that the breakdown point of the asymptotic estimator $\hat{\theta}_\infty$ depends only on the distribution.

Definition 2.1.2. *Consider a mixture distribution of $\Delta(x)$ with probability ϵ and $F(x)$ with probability $1-\epsilon$. Let $\hat{\theta}_\infty$ be the asymptotic estimator of θ, with parameter space $\boldsymbol{\Theta}$. The (asymptotic) breakdown point is the maximum ϵ^* so that*

$$\hat{\theta}_\infty((1-\epsilon)F + \epsilon\Delta) \cap \partial\boldsymbol{\Theta} = \emptyset \quad \forall\Delta(x), \forall\epsilon < \epsilon^*$$

which is the definition given for example in [5]. From now on, the asymptotic breakdown point will be referred simply as "breakdown point".

2.2 Measures of Location

2.2.1 M-Estimators

One of the most efficient way to estimate a parameter is by maximizing a given function of data and parameters, this is the key concept of the robust M-estimators for the location, first proposed by Huber in 1954 [6] as a generalization of the Maximum Likelihood Estimators.

Definition 2.2.1. *Let $\boldsymbol{X}$ be a set of random variables, let $\boldsymbol{\theta}$ be a set of parameters to be estimated. An M-estimation of $\boldsymbol{\theta}$ is the solution of the minimization problem*

$$\hat{\boldsymbol{\theta}} = arg\min_{\boldsymbol{\theta}} \left(\sum_i \rho(X_i, \boldsymbol{\theta}) \right)$$

Where $\rho(X_i, \boldsymbol{\theta}) : \Omega \to \mathbb{R}$ denotes a generic function of the random variables and parameters.

It can be seen that the standard MLE for i.i.d. variables can be obtained by choosing

$$\rho(X_i, \boldsymbol{\theta}) = -\log\left(f_{X|\boldsymbol{\theta}}(X_i|\boldsymbol{\theta})\right)$$

where $f_{X|\boldsymbol{\theta}}(X_i|\boldsymbol{\theta})$ is the probability density function of X with parameters $\boldsymbol{\theta}$.

This general definition can be, by choosing appropriate functions, applied successfully in robust statistics. The two most important functions used in literature will be presented.

Robust functions for M-estimators

A popular function for robust estimates is developed by Huber [6] in 1964.

Definition 2.2.2 (Huber's M-estimator)**.** *Let $\boldsymbol{X}$ be a set of random variables, let μ be a location parameter to be estimated. The Huber function for M-estimation of the location is defined, for each $X_i \in \boldsymbol{X}$ as*

$$\rho_H(X_i, \mu) = \begin{cases} (X_i - \mu)^2 & |X_i - \mu| \leq K \\ 2K|X_i - \mu| - K^2 & |X_i - \mu| > K \end{cases}$$

$$\psi_H(X_i, \mu) = \begin{cases} X_i - \mu & |X_i - \mu| \leq K \\ K \operatorname{sgn}(X_i - \mu) & |X_i - \mu| > K \end{cases} \tag{2.1}$$

Definition 2.2.3. *Let $\boldsymbol{X}$ be a set of random variables, let μ be a location parameter to be estimated. The Tukey's Bisquare function for M-estimation of the location is defined, for each $X_i \in \boldsymbol{X}$ as*

$$\rho_T(X_i, \mu) = \begin{cases} 1 - \left(1 - \left(\frac{X_i - \mu}{K}\right)^2\right)^3 & |X_i - \mu| \leq K \\ K^2 & |X_i - \mu| > K \end{cases}$$

$$\psi_T(X_i, \mu) = \begin{cases} (X_i - \mu)\left(1 - \left(\frac{X_i - \mu}{K}\right)^2\right)^2 & |X_i - \mu| \leq K \\ 0 & |X_i - \mu| > K \end{cases} \tag{2.2}$$

There is a non-differentiable biweight loss used in [7] which follows

$$\rho_B(X_i, \mu) = \begin{cases} (X_i - \mu)^2 & |X_i - \mu| \leq K \\ K^2 & |X_i - \mu| > K \end{cases} \tag{2.3}$$

2.2.2 L-Estimators

The main feature of the L-estimators is the use of a linear combination of sample quantiles in order to derivate a suitable estimation of the location, this approach is also suitable for highly variable distributions, even with unbounded first or second moments.

Sample Median

The simplest L-estimator is the sample median, defined for a sorted (ascending) sample $\boldsymbol{X}$ as it follows:

$$\tilde{X} = \begin{cases} X_{\frac{n+1}{2}} & \text{if } n \text{ odd} \\ \frac{1}{2}(X_{\frac{n}{2}} + X_{\frac{n}{2}+1}) & \text{if } n \text{ even} \end{cases}$$

The sample median has the advantage to be easy to calculate once the elements in a sample are sorted, furthermore it has a breakdown point of 50%, which makes it resistant in case of heavily contaminated distribution; it however presents some inefficiency for example under the hypothesis of a normal distribution of the sample. Let $f(x)$ be the sample's probability density function,then the asymptotic distribution of the sample median can be derived from the asymptotic distributions of the sample quantiles.

$$\tilde{X}_\infty \sim \mathcal{N}\left(m, \frac{1}{4nf^2(m)}\right) \tag{2.4}$$

where m is the median of the distribution $f(x)$.

Under the assumption of a normal distribution, $E[\tilde{X}_\infty] = m = \mu_X$, so the sample median is an unbiased estimator of the mean. Its efficiency, calculated as the inverse of its asymptotic variance, is lower than the sample mean estimator. Under normality its variance can be expressed as $\frac{\pi\sigma^2}{2n}$, thus, the sample variance has a relative efficiency of $\frac{2}{\pi} \approx 63.7\%$ with respect to the sample mean, which is the Best Linear Unbiased Estimator.

Trimmed Mean

In presence of a supposedly known percentage α of outliers in the sample it is possible to use the α-trimmed mean, which is the mean calculated after discarding the α lowest and the α highest observations.

2.3 Measures of Scale

A dispersion or scale estimator follows the following properties:

Shift invariance: if the sample is shifted by a constant c the dispersion estimator does not change

$$\theta(\boldsymbol{X} + c) = \theta(\boldsymbol{X}) \quad \forall c \in \mathbb{R} \tag{2.5}$$

Scale equivariance: if the sample is multiplied by a constant c the dispersion estimator will be multiplied by c too

$$\theta(c\boldsymbol{X}) = |c|\theta(\boldsymbol{X}) \quad \forall c \in \mathbb{R} \tag{2.6}$$

The most common example for a scale estimator is the standard deviation, calculated as the square root of the sample variance: in fact $\sqrt{S^2(\boldsymbol{X}+c)} = \sqrt{S^2(\boldsymbol{X})} = \hat{\sigma}$ and $\sqrt{S^2(c\boldsymbol{X})} = \sqrt{c^2 S^2(\boldsymbol{X})} = |c|\sqrt{S^2(\boldsymbol{X})} = |c|\hat{\sigma}$

2.3.1 Median Absolute Deviation

Definition 2.3.1. *The median absolute deviation (MAD) of a sample $\boldsymbol{X}$ is defined as the median of the absolute differences between $\boldsymbol{X}$ and its median.*

$$MAD(\boldsymbol{X}) = med|\boldsymbol{X} - med(\boldsymbol{X})|$$

Having in mind the properties of the median it is trivial to verify equations 2.5 and 2.6.

MAD is probably the most used robust dispersion estimator, the two main reasons why it is so popular are:

- MAD is computationally simple and efficient as it involves the calculation of two medians, this is an advantage especially for large datasets.
- MAD has a high breakdown point: since it involves only medians its breakdown point is equal to the median's one, 50%.

Before applying this estimator to data there are some things to consider.

Theorem 2.3.1. *Suppose that a sample $\boldsymbol{X}$ with cardinality $|\boldsymbol{X}| = N$ follows a normal distribution with mean μ and variance σ^2. Then*

$$\lim_{N\to\infty} E[MAD(\boldsymbol{X})] = \Phi^{-1}(0.75)\sigma \approx 0.675\sigma$$

The natural consequence is that MAD is a biased estimator of σ, this bias can be easily removed by correcting the sample MAD. Throughout the thesis the corrected MAD will be written as $MAD_c(.)$.

In R the function `mad` is already implemented with the bias correction and will be used throughout the thesis.

Furthermore it is not advised to use the MAD estimator if the distribution of the sample is suspected to be skewed (other measures like IQR are preferrable in this case).

MAD normal efficiency

Efficiency under normality for the MAD can be computed considering first the distribution of the deviation from the median in a normal distribution.

$$\boldsymbol{X} - med(\boldsymbol{X}) \sim \mathcal{N}\left(0, \frac{\pi\sigma^2}{2n} + \sigma^2\right) \tag{2.7}$$

Which is a sum of two independent normal RV. For sake of simplicity in notation $\frac{\pi\sigma^2}{2n}+\sigma^2$ will be defined as σ_d^2 Due to the fact that a normal distribution is symmetric around its median (and mean) it can be written that

$$P[|\boldsymbol{X} - med(\boldsymbol{X})| > a] = 2P[\boldsymbol{X} - med(\boldsymbol{X}) > a] \tag{2.8}$$

Finding the median of this absolute value is equal to find the quantile m so that

$$2P[\boldsymbol{X} - med(\boldsymbol{X}) > m] = 2P\left[Z > \frac{m}{\sigma_d}\right] = \frac{1}{2}$$

according to the standard normal table $m = 0.675\sigma_d$. By doing the limit for $N \to \infty$ the result of Theorem 2.3.1 is given. Its asymptotic efficiency can be calculated by recalling (2.4) and observing that the density of the absolute value of a normal with mean 0 is the double of a normal, only on positive values. So it can be written that

$$\lim_{N\to\infty} f_{AD}(m) = \frac{2}{\sigma\sqrt{2\pi}} \exp\left(-\frac{m^2}{2\sigma^2}\right) \approx \frac{\sqrt{2}}{\sigma\sqrt{\pi}} \exp\left(-\frac{0.675^2}{2}\right)$$

It is now possible to write the variance of the MAD

$$\lim_{N\to\infty} Var[MAD_c] \approx (0.675)^2 \frac{\pi\sigma^2}{8n} \exp\left(-\frac{0.675^2}{2}\right) \tag{2.9}$$

The efficiency of this estimator is calculated, with normality, to be 37% of the sample standard deviation. With this in mind, looking for more efficient estimators can be useful in order to get more accurate values for the dispersion parameter in a dataset with outliers.

2.3.2 Efficient alternatives to MAD

Although the MAD brings several advantages it might be useful to introduce two more efficient estimators that have, as MAD the maximum breakdown point possible, which is 50%. These estimators are implemented efficiently in the `R` package `robustbase` [8] [9].

Q-Estimator

Rousseeuw and Croux [10] came up with two main alternatives to MAD, the most used of which is the Q-Estimator, which has 82% of efficiency under normality.

Definition 2.3.2 (Rousseeuw and Croux's Qn)**.** *Given a sample* $\boldsymbol{X}$ *the robust scale estimator* Q_n *is defined as*

$$Q_n(\boldsymbol{X}) = C\left\{|X_i - X_j|; i < j\right\}_{(k)}$$

Where

$$k = \binom{\left\lfloor \frac{n}{2} \right\rfloor + 1}{2}$$

the notation $\{.\}_{(k)}$ *denotes the k-th order statistic and* C *is a constant for consistency.*

In order to get some idea of its computational time the estimate of the scale parameter of the vector `res` defined in is performed

```
system.time(Qn(res))
```

```
##    user  system elapsed
##    0.08    0.00    0.08
```

τ-Estimator

The location-scale robust estimator τ_s is introduced and defined by Yohai and Zamar [11]. In order to better understand how τ_s is made two functions are introduced.

$$W_1(x,c) = \left(1 - \left(\frac{x}{c}\right)^2\right)^2 \mathbb{1}(|x| < c); \qquad W_2(x,c) = \min(x^2, c^2)$$

It is interesting to note that the function W_1 resembles the Tukey's Bisquare mentioned in while W_2 resembles the Biweight estimator.

Definition 2.3.3 (Yohai and Zamar)**.** *Given a sample* $\boldsymbol{X}$ *with sample size* $N = |\boldsymbol{X}|$ *the robust scale estimator* τ_s *is defined as*

$$\tau_s(\boldsymbol{X})^2 = \frac{MAD(\boldsymbol{X})^2}{N} \sum_{i=1}^{N} W_2\left(\frac{X_i - \mu_\tau(X)}{MAD(\boldsymbol{X})}, c_2\right)$$

Where μ_τ *is an estimate of the location parameter defined starting from the weights*

$$w_i = W_1\left(\frac{X_i - Med(\boldsymbol{X})}{MAD(\boldsymbol{X})}, c_1\right) \qquad as \qquad \mu_\tau = \frac{\sum_{i=1}^{N} w_i X_i}{\sum_{i=1}^{N} w_i}$$

The constants $c_1 = 4.5$ and $c_2 = 3$ are chosen according to Maronna and Zamar [12] (section 3), so that they "yield approximately 80% efficient univariate location and scale for both normal and Cauchy data", which is comparable to the Q_n estimator seen previously.

In order to get some idea of its computational time the same estimate of the scale parameter as Q_n is performed

```
system.time(scaleTau2(res))
```

```
##    user  system elapsed
##       0       0       0
```

This suggest the preferrable use of the τ_s estimator when dealing with large datasets.

2.4 Robust Autocorrelation

The last measure to be presented in some robust alternatives is the correlation, which further in the thesis is going to be used when calculating the autocorrelation plot of a time series with outliers.

The standard measure of correlation, by Pearson, uses sample variances which, as stated before, have a breakpoint of 0%, thus can lead to misleading results even in presence of a single outlier.

2.4.1 Spearman's ACF

Spearman [13] proposed a non-parametric alternative to the Pearson's correlation coefficient, this makes it possible to define a robust, non-parametric auto-correlation which will be useful to analyze time series with outliers.

Definition 2.4.1 (Spearman's rho)**.** *Let Y_t be a continuous time-series, let r a be a function that converts observations in a sample into ranks. If no ties are present*

$$r(Y_i) = \sum_j \mathbb{1}\{Y_j > Y_i\}$$

Spearman's autocorrelation at lag k is defined as the Pearson's autocorrelation at lag k performed on ranks

$$\rho_s(Y_t, k) = \frac{\hat{Cov}[r(Y_t), r(Y_{t-k})]}{\hat{Var}[r(Y_t)]}$$

Where $\hat{Cov}$ is the sample covariance and $\hat{Var}$ is the sample variance of the ranks.

If no ties are present (which is likely in samples from a continuous distribution) there is a fast way to calculate $\rho_s(Y_t, k)$

$$1 - \frac{6 \sum_{i=k+1}^{N} [r(Y_i) - r(Y_{i-k})]}{N(N^2 - 1)}$$

Which is the formula used by any statistical software.

```
system.time(acfrob(res,approach="rank",plot=F))
```

```
##    user  system elapsed
##    0.04    0.00    0.12
```

The rank correlation is easily computable and fast, however its breakdown point is not trivial and it is well explained by Davies and Gather [14].

2.4.2 GK Estimator

The GK estimator for autocorrelation is based on the covariance estimator by Gnanadesikan and Kettenring [15] which in 1972 used the identity

$$Cov(X, Y) = \frac{1}{4} \left(Var[X + Y] - Var[X - Y] \right)$$

to relate a robust covariance function to the calculation of robust variance functions.

Ma and Genton in 2000 [16] revisited this approach in order to build an autocorrelation function one lag at a time.

Definition 2.4.2 (GK autocorrelation)**.** *Let Z_t be a time series with $N = |Z_t|$. In order to find its correlation at lag i, let $X = \{Z_j : j > i\}$ and $Y = \{Z_j : j < N - i\}$. Let $S_r^2(.)$ be a robust estimator of the variance, then the GK autocorrelation is defined as*

$$\rho_{GK}(i) = \frac{S_r^2(X+Y) - S_r^2(X-Y)}{S_r^2(X+Y) + S_r^2(X-Y)}$$

The breakdown point of this statistic is at most 25%, since an outlier in Z_t may appear twice in both $X + Y$ and $X - Y$ and the maximum breakdown point for an estimator is, as previously said, 50%.

Now a test on the computational time of this robust autocorrelation is performed.

```
system.time(acfrob(res,approach="GK",scalefn = scaleTau2,plot=F))
```

```
##    user  system elapsed
##    7.55    0.23    8.04
```

The computation of this statistic is much more expensive, even when using a fast estimator like τ_s. However it is recommended when dealing with a significantly contaminated dataset ($\approx$ 20% of outliers).

Chapter 3

A Manufacturing System Model with Measurement Delays

This short chapter will focus on a statistical model of negative correlated inter-event times in a manufacturing process composed by one station, providing an explanation for the negative correlation that may arise in some situations.

3.1 Model with independent IETs

When dealing with a single station in a manufacturing process, the main hypothesis that can be made is that, as long as the machine is on there are always raw materials ready to be processed. This means that the station is modeled as a degenerate G/GI/1 queue where the idle time for any piece is exactly 0, the service times $\boldsymbol{X}$ are independent, and the departure process has independent IETs which are equal to the service times.

The departure process is modeled as a counting process with an increment when the piece i is produced; when this event occurs the related timestamp is registered.

A set of random variables $\boldsymbol{Y}$ is defined where Y_i is the time where the i-th new piece is produced since the machine has been turned on and begun the actual production process. The simplest model, where the service times are i.i.d., states

that

$$P_i = \sum_i X_i \qquad supp\{f_X(x)\} \subseteq [0, \infty) \tag{3.1}$$

so, by defining $\boldsymbol{Y}$ as the set of the inter-event times

$$Y_i = P_i - P_{i-1} = X_i \tag{3.2}$$

In order to characterize the distribution of X_i some assumptions have to be made about the type of distribution, then its parameters can be estimated by maximizing the likelihood function.

Even in presence of change-points and outliers, if equation 3.2 holds it is possible to obtain a correct estimate of the scale parameter using robust estimators and some properties of the differences of random variables.

Let $Y_t^- = \{Y_2 - Y_1, \ldots, Y_T - Y_{T-1}\}$ be the (ordered) vector of consecutive differences for Y_t. Then

$$Y_i^- = \begin{cases} \mu_{j+1} - \mu_j + \eta_{i+1} - \eta_i & \text{if} \quad i \subseteq \boldsymbol{\tau} \\ \eta_{i+1} - \eta_i & \text{otherwise} \end{cases}$$

This means that $E[Y_i^- | i \subseteq \boldsymbol{\tau}] = \mu_{j+1} - \mu_j$ and $E[Y_i^- | i \nsubseteq \boldsymbol{\tau}] = 0$, having that $P[i \subseteq \boldsymbol{\tau}] = \frac{n}{T}$. This also means that $Var[Y_i^-] = 2Var[\eta] = 2\sigma^2 \quad \forall i$.

Since Y_t^- follows a contaminated distribution it is possible to estimate σ by choosing $\hat{S}_r(Y_t^-)$, a consistent robust estimator of $\sigma_{Y_t^-}$ with breakdown point higher than $\frac{1}{T}(n + 2n_\omega)$.

$$\hat{\sigma} = \frac{\hat{S}_r\left(Y_t^-\right)}{\sqrt{2}} \tag{3.3}$$

Where n_ω is the number of outliers present in Y_t, since a single outlier in Y_i has a role in observations Y_i^- and Y_{i+1}^-.

Since n and n_ω are unknown it is recommended to use an estimator with the highest possible breakdown point (50%), such as MAD_c. This explains the estimate of the scale parameter `sigmah` in Section 1.4, since the elements in Y_t were assumed to be independent.

3.2 Model with non negligible delays

When dealing with processes with low variability, a negative correlation between the inter-event times can happen, apparently breaking the hypothesis of independence for the service times. However, in formulating equation 3.1, a strong hypothesis has been made: the effect of any measurement error is negligible, so that it can be set to zero. If a negative correlation is present it is possible that this hypothesis is not valid, in order to prove it the measurement errors are modeled with a set of i.i.d. random variables $\boldsymbol{\varepsilon}$ with finite mean and variance.

$$P_i = \sum_i X_i + \varepsilon_i \qquad supp\{f_X(x)\}, supp\{f_\varepsilon(e)\} \subseteq [0, \infty) \tag{3.4}$$

If this model is correct the inter-event times will be expressed as follows

$$Y_i = P_i - P_{i-1} = X_i + \varepsilon_i - \varepsilon_{i-1} \tag{3.5}$$

It can be seen that $E[Y] = E[X]$ and $Var[\Delta] = 2Var[\varepsilon] + Var[X]$

Calculating the autocovariance of lag 1 of the inter-event times, supposing $\boldsymbol{X}$ and $\boldsymbol{\varepsilon}$ independent

$$Cov[Y_i, Y_{i-1}] = Cov[X_i + \varepsilon_i - \varepsilon_{i-1}, X_{i-1} + \varepsilon_{i-1} - \varepsilon_{i-2}] = -Var[\varepsilon_i]$$

So the autocorrelation at lag 1 becomes

$$\rho_1(Y) = -\frac{Var[\varepsilon]}{Var[Y]} = -\frac{Var[\varepsilon]}{2Var[\varepsilon] + Var[X]}$$

It can be of interest to see that the estimate as written in (3.3) is biased.

First it can be observed that, if (3.5) is valid, then, for any i not affected by outliers or change-points

$$Y_i^- = \eta_{i+1} - \eta_i + \varepsilon_{i+1} - 2\varepsilon_i + \varepsilon_{i-1}$$

Meaning that $Var[Y_i^-] = 2\sigma^2 + 6\sigma_\varepsilon^2$.

If $\hat{S}_r(Y_t^-)$ is, as mentioned above, a robust, consistent estimator of $\sigma_{Y_t^-}$, then

$$E\left[\hat{S}_r\left(\frac{Y^-}{\sqrt{2}}\right)\right] = \sqrt{\sigma^2 + 3\sigma_\varepsilon^2} \neq \sqrt{\sigma^2 + 2\sigma_\varepsilon^2} \tag{3.6}$$

Which proves that the estimate is biased, hence not correct.

Note that when $Var[X] \gg Var[\varepsilon]$ the model can be simplified to the one in equation 3.1, regardless of $E[\varepsilon]$ which can assume (in theory) any value.

The model can be difficult to deal with, since the fact that measurement delays are present creates dependence between the observations in the same on-off cycles. Mind that the fact that in (3.5) there is a dependence only to ε_{i-1} does not make the variable Y_i dependent only on Y_{i-1}: the variable Y_i, given that the machine turns on before Y_1 is dependent on $\boldsymbol{Y}_{(1:i)}$. The computation of the likelihood for this model can be complex, since the calculation of the joint distribution involves an integration on $\boldsymbol{\varepsilon}$, however the next section presents an approximated case where the integration can be simplified, then computed numerically.

3.3 Model with Normal Approximations

There are some processes with extremely low variability so that the inter-event times can be approximated to a distribution with support in $\mathbb{R}$: as an example if the X_i follow a bell-shaped distribution (i.e. Gamma) with $E[X] = 10\sqrt{Var[X]}$ the probability for the moment-fitted normal to go below zero is roughly $7.62 \cdot 10^{-24}$, which is totally negligible.

In this section the model, starting from equation 3.4 and from a normal approximation of the variables $\boldsymbol{X}$ and $\boldsymbol{\varepsilon}$ is studied. It is important to know that this approximation does not always apply since, in order to be as close as possible to a normal the conditions $\mu \gg \sigma$ and $\mu_\varepsilon \gg \sigma_\varepsilon$ have to be respected.

A density plot of the centered data (variable `res` in section 1.4) is drawn in figure 3.1 which shows a normal-like behaviour. The model can be applied since the mean never goes below 20 seconds while the standard deviation is in the order of 10^{-1} seconds.

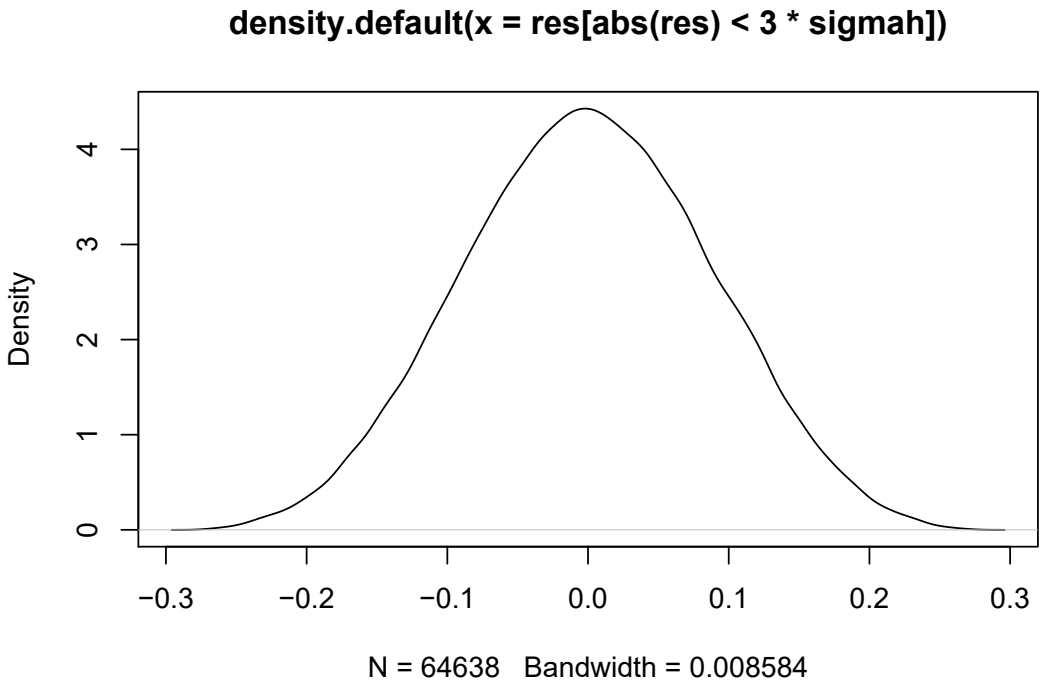

Figure 3.1: Density plot of the centered data (no outliers)

Treating these positive distributions as normal has to be done carefully, however the model based on it has a huge advantage: it is possible to find an efficiently computable likelihood.

Theorem 3.3.1. *Let $\boldsymbol{Y}$ be a set of random variables which can be written as in equation 3.5 where $X_i \sim \mathcal{N}(\mu_i, \sigma^2)$ and $\varepsilon_i \sim \mathcal{N}(\mu_\varepsilon, \sigma_\varepsilon^2)$. The log-likelihood of $\boldsymbol{Y}$ assumes the form*

$$\ell(\boldsymbol{Y};\boldsymbol{\theta}) = -\frac{2N+1}{2}\log(2) - \frac{N}{2}\log(\pi\sigma^2) - \frac{N+1}{2}\log(\sigma_\varepsilon^2) - \frac{1}{2}\sum_{i=0}^{N}\log(a_i) - \sum_{i=0}^{N+1} c_i$$

where

$$c_0 = \sum_{i=1}^{N}\frac{(Y_i-\mu_i)^2}{2\sigma^2}, \qquad c_i = -\frac{b_{i-1}^2}{4a_{i-1}} \quad \forall i \in \{1...N+1\}$$

and $\{a_i, b_i\}$ is a system of successive recursions.

Proof. In order to prove this theorem an expression for the total likelihood function of $\boldsymbol{Y}$ has to be found.

By the law of total probability for continuous distributions

$$f(\boldsymbol{Y}|\boldsymbol{\theta}) = \int_{\Omega_\varepsilon^{N+1}} f(\boldsymbol{Y}|\boldsymbol{\varepsilon}, \boldsymbol{\theta}) df(\boldsymbol{\varepsilon}|\boldsymbol{\theta})$$

Where Ω_ε is the support of any $\varepsilon \in \boldsymbol{\varepsilon}$.

By conditioning on $\boldsymbol{\varepsilon}$, it can be observed that every Y_i is independent to each other and follows

$$Y_i|\boldsymbol{\varepsilon} \sim \mathcal{N}(\mu + \varepsilon_i - \varepsilon_{i-1}, \sigma^2) \tag{3.7}$$

So the likelihood function can be expressed as an N+1-dimensional integral

$$f(\boldsymbol{Y}|\boldsymbol{\theta}) = C \int_{\mathbb{R}^{N+1}} \exp\left[-\sum_{i=1}^{N} \frac{(\xi_i - \varepsilon_i + \varepsilon_{i-1})^2}{2\sigma^2} - \sum_{i=0}^{N} \frac{(\varepsilon_i - \mu_\varepsilon)^2}{2\sigma_\varepsilon^2} \right] d\boldsymbol{\varepsilon} \tag{3.8}$$

where $\xi_i = x_i - \mu_i$ an $C = \left((2\pi)^{2N+1}(\sigma^2)^N(\sigma_\varepsilon^2)^{N+1}\right)^{-\frac{1}{2}}$.

The first thing to see when writing the probability density function of $\boldsymbol{Y}$ is that the parameter μ_ε is not identifiable.

It can be in fact observed that, for any value of μ_ε, the substitution $\varepsilon_i^* = \varepsilon_i - \mu_\varepsilon$ in the integral can be done for all i; this results in

$$\frac{1}{\sqrt{(2\pi)^{2N+1}(\sigma^2)^N(\sigma_\varepsilon^2)^{N+1}}} \int_{\mathbb{R}^{N+1}} \exp\left[-\sum_{i=1}^{N} \frac{(\xi_i - \varepsilon_i^* + \varepsilon_{i-1}^*)^2}{2\sigma^2} - \sum_{i=0}^{N} \frac{\varepsilon_i^{*2}}{2\sigma_\varepsilon^2} \right] d\boldsymbol{\varepsilon}^*$$

This means that any choice of μ_ε leads to the same likelihood function, hence the parameter is not identifiable.

In order to simplify some calculations, the integral will be solved by putting $\mu_\varepsilon = 0$, the result will be the same for all μ_ε.

Developing the squares, (3.8) will become

$$C \int_{\mathbb{R}^{N+1}} \exp\left[-\left(\sum_{i=1}^{N} \frac{\xi_i^2}{2\sigma^2} + \sum_{i=1}^{N} \frac{\varepsilon_i^2}{2\sigma^2} + \sum_{i=0}^{N-1} \frac{\varepsilon_i^2}{2\sigma^2} - \sum_{i=1}^{N} \frac{\varepsilon_i \varepsilon_{i-1}}{\sigma^2} - \sum_{i=1}^{N} \frac{\xi_i \varepsilon_i}{\sigma^2} + \right.\right.$$
$$\left.\left. + \sum_{i=1}^{N} \frac{\xi_i \varepsilon_{i-1}}{\sigma^2} + \sum_{i=0}^{N} \frac{\varepsilon_i^2}{2\sigma_\varepsilon^2} \right)\right] d\boldsymbol{\varepsilon}$$

now the following quantities are defined.

$$\alpha_N = \alpha_0 = \frac{1}{2\sigma^2} + \frac{1}{2\sigma_\varepsilon^2}; \qquad \alpha_i = \frac{1}{\sigma^2} + \frac{1}{2\sigma_\varepsilon^2} \quad \forall i \in \{1 \dots N-1\}$$

$$\beta_0 = \frac{\xi_1}{\sigma^2}; \qquad \beta_N = -\frac{\xi_N}{\sigma^2}$$

$$\beta_i = \frac{\xi_{i+1} - \xi_i}{\sigma^2} = \quad \forall i \in \{1 \dots N-1\}$$

$$\gamma = -\frac{1}{\sigma^2}$$

$$c_0 = \sum_{i=1}^{N} \frac{\xi_i^2}{2\sigma^2}$$

So that the likelihood can be written as

$$C \int_{\mathbb{R}^{N+1}} \exp\left[-\left(\sum_{i=0}^{N} \alpha_i \varepsilon_i^2 + \sum_{i=0}^{N} \beta_i \varepsilon_i + \sum_{i=0}^{N-1} \gamma \varepsilon_i \varepsilon_{i+1} + c_0 \right) \right] d\boldsymbol{\varepsilon}$$

Now every term that involves ε_0 is isolated

$$C e^{-c_0} \int_{\mathbb{R}^N} \exp\left[-\left(\sum_{i=1}^{N} \alpha_i \varepsilon_i^2 + \sum_{i=1}^{N} \beta_i \varepsilon_i + \sum_{i=1}^{N-1} \gamma \varepsilon_i \varepsilon_{i+1} \right) \right] \int_{-\infty}^{\infty} I_0 d\varepsilon_0 d\boldsymbol{\varepsilon}_{(1:N)}$$

having defined

$$I_0 = \exp\left\{ -\left[a_0 \varepsilon_0^2 + (b_0 + \gamma \varepsilon_1) \varepsilon_0 \right] \right\}$$

where $a_0 = \alpha_0$ and $b_0 = \beta_0$.

By completing the square the integral can be solved

$$\int_{-\infty}^{\infty} I_0 d\varepsilon_0 = \exp\left(\frac{(b_0 + \gamma \varepsilon_1)^2}{4a_0} \right) \sqrt{\frac{\pi}{a_0}}$$

so that the likelihood becomes

$$C\sqrt{\frac{\pi}{a_0}}e^{-c_0-c_1}\int_{\mathbb{R}^{N-1}}\exp\left[-\left(\sum_{i=2}^{N}\alpha_i\varepsilon_i^2+\sum_{i=2}^{N}\beta_i\varepsilon_i+\sum_{i=2}^{N-1}\gamma\varepsilon_i\varepsilon_{i+1}\right)\right]\int_{-\infty}^{\infty}I_1 d\varepsilon_1 d\boldsymbol{\varepsilon}_{(2:N)}$$

with

$$c_1=\frac{-b_0^2}{4a_0};\qquad I_1=\exp\left\{-\left[a_1\varepsilon_1^2+(b_1+\gamma\varepsilon_2)\varepsilon_1\right]\right\}$$

where

$$a_1=\alpha_1-\frac{\gamma^2}{4a_0};\qquad b_1=\beta_1-\frac{\gamma b_0}{2a_0}$$

By grouping terms and integrating, it is possible to obtain recursive sequences for a_i and b_i.

$$a_i=\alpha_i-\frac{\gamma^2}{4a_{i-1}} \tag{3.9}$$

$$b_i=\beta_i-\frac{\gamma b_{i-1}}{2a_{i-1}} \tag{3.10}$$

$$c_i=-\frac{b_{i-1}^2}{4a_{i-1}} \tag{3.11}$$

The initial conditions are $a_0=\alpha_0$ and $b_0=\beta_0$ as stated previously.

After N+1 integrations the likelihood becomes

$$\begin{aligned}\boldsymbol{\mathcal{L}}(\boldsymbol{\theta};\boldsymbol{Y})&=C\sqrt{\frac{\pi^{N+1}}{\prod_{i=0}^{N}a_i}}\exp\left(-\sum_{i=0}^{N+1}c_i\right)\\&=\frac{1}{\sqrt{2^{(2N+1)}(\pi\sigma^2)^N(\sigma_\varepsilon^2)^{N+1}\prod_{i=0}^{N}a_i}}\exp\left(-\sum_{i=0}^{N+1}c_i\right)\end{aligned} \tag{3.12}$$

Then the log-likelihood can be computed as

$$\ell(\boldsymbol{\theta};\boldsymbol{Y})=-\frac{2N+1}{2}\log(2)-\frac{N}{2}\log(\pi\sigma^2)-\frac{N+1}{2}\log(\sigma_\varepsilon^2)-\frac{1}{2}\sum_{i=0}^{N}\log(a_i)-\sum_{i=0}^{N+1}c_i. \tag{3.13}$$

The proof is concluded. □

In order to find the ML estimates $\hat{\boldsymbol{\mu}}$, $\hat{\sigma}^2$ and $\hat{\sigma}_\varepsilon^2$, the log-likelihood in this form involves a system of recursive successions instead of an N+1-dimensional integral, thus it can be easily maximized numerically with functions like `nlm` (for which it is advised to log-transform the variances since it is an unconstrained minimizer).

The model, despite its approximations, will be used in the next chapters in order to select the best change-point model: the approximated likelihood captures the negative autocorrelation at lag 1 with the introduction of the parameter σ_ε^2, making it a fast and more reliable (with respect to the models that assume independence between the inter-event times) way to measure how well a change-point model fits to this type of data.

Chapter 4

Change-Point Analysis

4.1 Modeling change-points in mean

Given a Time-Series $Y_t = \{Y_1, Y_2, \ldots, Y_T\}$ a number n of change-points, an ordered vector of change-point locations $\boldsymbol{\tau} : \tau_{j+1} > \tau_j \quad \forall j \in \{1, \ldots, n-1\}$ and a set of $n+1$ probability density functions $\mathbf{p} = \{p_1(y), ..., p_{n+1}(y)\}$ are assumed.

Change-point models are a class for which the following condition is respected:

$$f_{Y_i}(y) = p_j(y) \iff \tau_j^c < i \leq \tau_{j+1}^c \tag{4.1}$$

Where $\boldsymbol{\tau}^c = \{0, \boldsymbol{\tau}, T\}$.

The main objectives, given this class of models are:

1. A correct estimate $\hat{n}$ of the number of change-points
2. A correct estimate $\hat{\boldsymbol{\tau}}$ of the location of these change-points
3. A correct estimate $\hat{\mathbf{p}}(y)$ of the probability density functions.

A frequent hypothesis in change-point models is that the probability densities in $\mathbf{p}$ come from the same class of distributions (i.e. normal, gamma) and the only difference between them is represented by one or more parameters.

The focus in this chapter will be on optimal detection of change-points in the expected value of the distribution $p(y)$, so that the model in (4.1) becomes

$$Y_i = \mu_j + \eta_i \iff \tau_j^c < i \leq \tau_{j+1}^c \tag{4.2}$$

where $\boldsymbol{\mu}$ is a vector of real numbers and $\boldsymbol{\eta}$ is a set of identically distributed random variables with mean 0. Various change-point in mean algorithms are applied to the manufacturing process cycle times and compared in order to determine the most suitable segmentation for that type of data. The algorithms presented involve functions and statistics that are resistant to outliers, seen in chapter 2.

4.2 Robust Loss Function Minimization

The change-point problem can be formulated as a minimization of a well-chosen loss function.

In order to choose a loss function for the whole data it is recommended to define one for a segment.

The main option is the quadratic loss for a given segment

$$\xi(Y_t, \mu_j, \tau_j^c, \tau_{j+1}^c) = \sum_{i=\tau_j^c+1}^{\tau_{j+1}^c} (Y_i - \mu_j)^2 \tag{4.3}$$

In this case minimizing μ_j is the equivalent of doing a Maximum Likelihood Estimation of the mean for each segment, assuming i.i.d. normal distributions. Dealing exclusively with shift in the mean with constant variance, the minimization of the sum over all segments of the minimized quadratic losses will be equal to the MLE of the time series following this model. In order to prevent overfitting a constant penalty is added for every change-point added to the model.

$$\mathcal{C}(Y_t, \boldsymbol{\tau}) = \sum_j \left(\min_{\mu_j} \xi(Y_t, \mu_j, \tau_j^c, \tau_{j+1}^c) + \beta \right) \tag{4.4}$$

The penalty β can be viewed in this case as a model selection criterion, in fact the most popular choices of it are the ones which relate to some information-based (AIC or BIC) or some other known model selection criterion.

When dealing with outliers in the data, the quadratic cost as described in (4.3) is not reliable, as seen in Figure 1.8.

Given a time series Y_t with one change-point τ and one outlier Y_k, there is a value of Y_k for which, using the quadratic loss minimizing it isn't possible to find τ.

Suppose that $k < \tau$, if $Y_k = (T - \tau)\mu_2 - (\tau - 1)\mu_1$ then the expected values of the sample means become

$$E[\hat{\mu}_1] = E[Y_{1:\tau}^{(-k)}] + E[Y_k] = (\tau-1)\mu_1 + (T-\tau)\mu_2 - (\tau-1)\mu_1 = (T-\tau)\mu_2 = E[\hat{\mu}_2]$$

When it comes to the minimization of the loss function, the introduction of a new change-point is very unlikely if the penalty β is properly chosen. In order to avoid this type of problems the introduction of new robust estimators is needed.

A general formula for the loss function of a segment can be

$$\xi(Y_t, \mu_j, \tau_j^c, \tau_{j+1}^c) = \sum_{i=\tau_j^c+1}^{\tau_{j+1}^c} \gamma(Y_i, \mu_j)$$

The choice for $\gamma(Y_i, \mu_j)$ has to be made so that outliers do not affect the change-point model. A function that mitigates the presence of extreme values is the Huber Loss (see Definition 2.2.2) . Huber's loss is unbounded, therefore it is not resistant to arbitrarily high or low values in Y_t.

The algorithm used to minimize the loss function is the R-FPOP, created by Fearnhead and Rigaill [7] as a robust version of the Functional Pruning Optimal Partitioning (FPOP) by Maidstone, Hocking ,Fearnhead, Rigaill [17].

The R-FPOP algorithm is implemented on the `R` function `Rob_seg.std` by Rigaill in the package `robseg` [3].

A model using Huber loss is applied to the data using the recommended `lthreshold`.

```
cpm.hub<-Rob_seg.std(s_,loss="Huber"
                     ,lambda=2*log(n)
                     ,lthreshold=1.345)
cpm.hub$K
```

```
## [1] 160
```

The number of change points is less than what found using a non-robust algorithm. However there is a huge risk that some segments are formed by only one element. It can be seen, using the estimated change-point locations how many segments have length equal to one.

```
sum(diff(c(0,cpm.hub$t.est))==1)
```

```
## [1] 73
```

This quantity, in a robust model has to be equal to zero.

```
#plotting the huber loss model
plot(s,ylim=c(20,60))
abline(v=cpm.hub$t.est[-cpm.hub$K],col="blue")
```

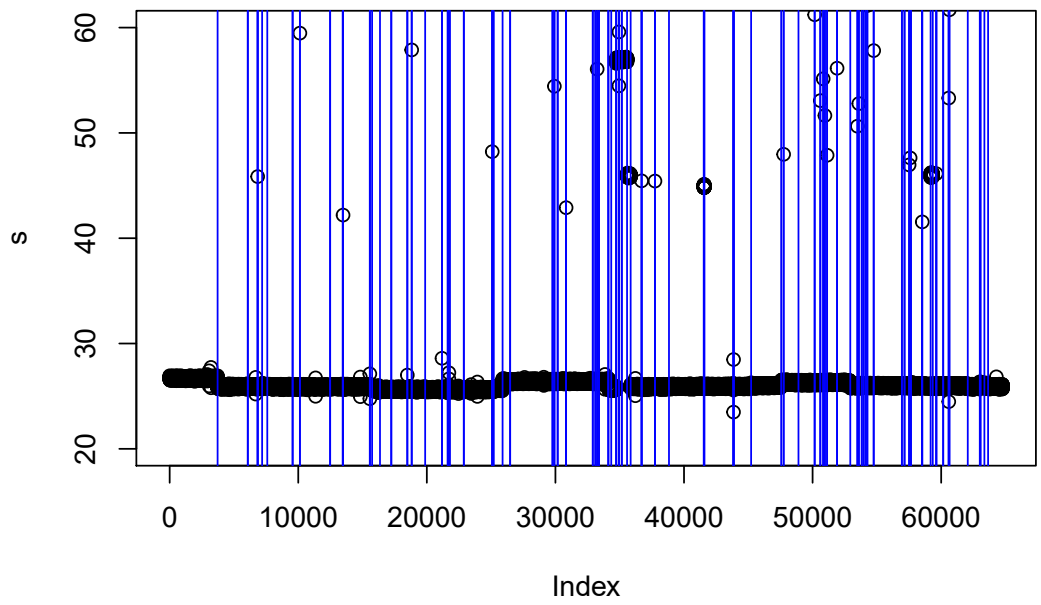

Figure 4.1: Change Points using Huber's Loss

The plot in figure 4.1 tells that the Huber loss is not robust enough, as it classifies some outliers as segments. This behaviour can be explained by the fact that, as mentioned in section 2.2.1, this loss function is unbounded, therefore it is not resistant to arbitrarily high or low values in Y_t.

In case of extreme values in the time series it is advised to use a bounded function like the Biweight Loss ρ_B as in (2.3). The parameter `lthreshold` for the algorithm is set to 3 as recommended in [7], meaning that $K = 3\hat{\sigma}$.

```
cpm.biw.bic<-Rob_seg.std(s_,loss="Outlier"
            ,lambda=2*log(n)
            ,lthreshold=3)
cpm.biw.bic$K
```

```
## [1] 40
```

```
plot(s,ylim=c(20,60))
lines(cpm.biw.bic$smt*sigmah,col="red")
abline(v=cpm.biw.bic$t.est[-cpm.biw.bic$K],col="blue")
```

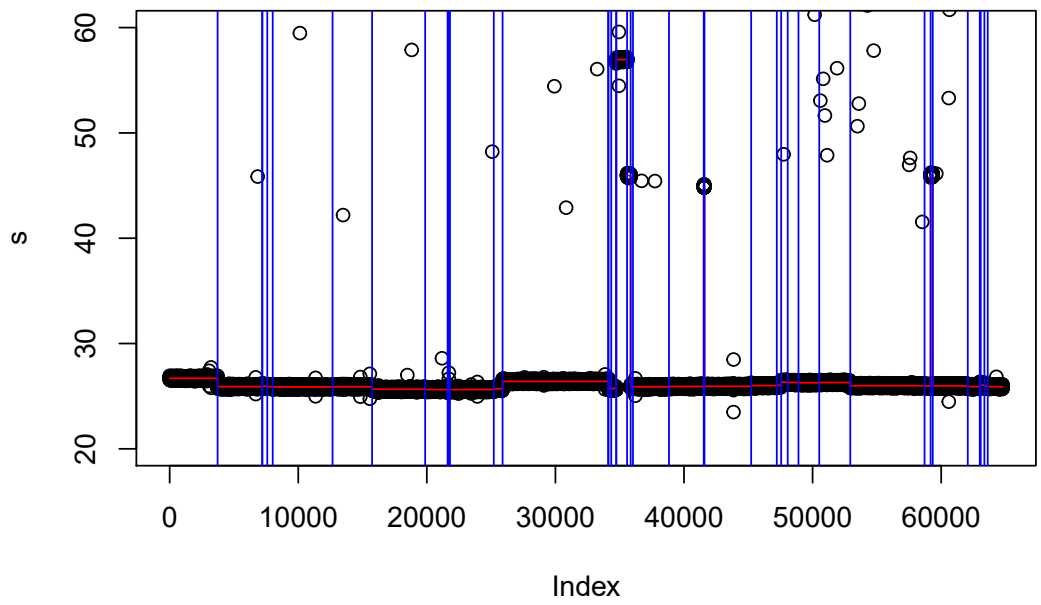

Figure 4.2: Change-points using Biweight Loss (penalty: 2log(N))

The number of change-points is much lower than what found in Huber's loss and figure 4.2 shows that (most of) the outliers do not affect the change-point model. This result can be explained by theorem 2 in [7] which states that the minimum segment length has a lower bound which is function of the parameters `lambda` and `lthreshold`. In this case

$$\text{msl} = \left\lceil \frac{2\log(N)}{3^2} \right\rceil = 3$$

It can be of interest to compute the length of the shortest segment, given this model.

```
data.frame(single.segs=sum(diff(c(0,cpm.biw.bic$t.est))==1),
min.dist=min(diff(cpm.biw.bic$t.est)))
```

```
##   single.segs min.dist
## 1           0        7
```

As expected there are no segments of length 1. The smallest segment, which has

length equal to 7, is very likely to be a group of outliers. The segmentation is clearly not optimal and there is no information criterion that justifies the choice of the penalty, hence it has to be improved and the best way to do it is by applying an efficient algorithm that helps tuning the penalty parameter.

4.2.1 Robust CROPS

Change-point over a Range Of Penalties (CROPS) is an algorithm aimed to detect all the optimal segmentations within a chosen penalty range $[\beta_0, \beta_1]$ in a change-point problem solved with loss minimization. The algorithm is created and developed by Haynes, Eckley and Fearnhead [18] and is based on two key concepts:

- The number of change-points found is a monotonically decreasing function of the penalty.
- if two different penalties return the same number of change-points, then the change-point locations have to be the same.

The `crops` function used in the following code chunk is created by following Algorithm 2 in [18] using the robust FPOP algorithm with biweight loss, via the function `rob_seg.std`, instead of PELT. The input parameters are respectively the data (in this case scaled by its standard deviation estimate), the minimum and the maximum of the penalty range and the parameter `K` of the biweight loss.

```
crops.biw<-crops(s_,2*log(n),50*log(n),3)
crops.biw<-crops.biw[order(crops.biw$ncp),]
crops.biw<-aggregate(crops.biw,by=list(crops.biw$ncp),FUN = min)[,-1]
crops.biw$beta.log<-crops.biw$beta/log(n)
crops.biw<-cbind(crops.biw,msl=sapply(crops.biw$beta,
function(b) min(diff(c(0,Rob_seg.std(s_,
loss="Outlier",lambda=b,lthreshold=3)$t.est)))))
crops.biw<-crops.biw[crops.biw$msl>=30,]
head(crops.biw)
```

```
##   ncp     beta     cost beta.log msl
## 1  14 361.0536 65785.57 32.58651 166
## 2  16 354.3906 65063.46 31.98515 166
## 3  18 342.2343 64368.00 30.88799  58
## 4  19 317.9216 64028.45 28.69368  58
## 5  20 175.1829 63732.16 15.81095  58
## 6  22 165.3832 63381.80 14.92648  58
```

The resulting data frame has five variables:

- `ncp` is the number of change-points found

- `beta` is the penalty required to find the number of change-points
- `beta.log` is the penalty divided by the logarithm of `n`
- `cost` is the total cost function minimized by the robust FPOP algorithm
- `msl` is the minimum length of a segment. Penalties for which the value of `msl` is less than 30 are discarded

If the data were independent the cost function could be useful in order to detect the optimal segmentation through a scree plot. Instead the penalties found via the CROPS algorithm are used to find the approximate log-likelihood given a number of change-points.

```
suppressWarnings(
for(i in 1:nrow(crops.biw)){

  cpts<-c(0,Rob_seg.std(s_,loss="Outlier"
                  ,lambda=crops.biw$beta[i]
                  ,lthreshold=lth)$t.est)

  meds=sapply(1:(crops.biw$ncp[i])
              ,function(j) median(s[(cpts[j]+1):cpts[j+1]]))

  wps<-model.cp.wp(mu=meds
                  ,sigmah^2/2,sigmah^2/2)

  cpm.loss.biw.ll[i]=-nlm(mlltot,wps,x=df.ll,cps=cpts[-1])$minimum
})
```

Now that the approximated log-likelihoods are computed and maximized a scree plot can be done in order to inspect how this likelihood evolves in the various models.

```
plot(crops.biw$ncp,cpm.loss.biw.ll
     ,xlab="Number of Changepoints"
     ,ylab="LogLikelihood")
```

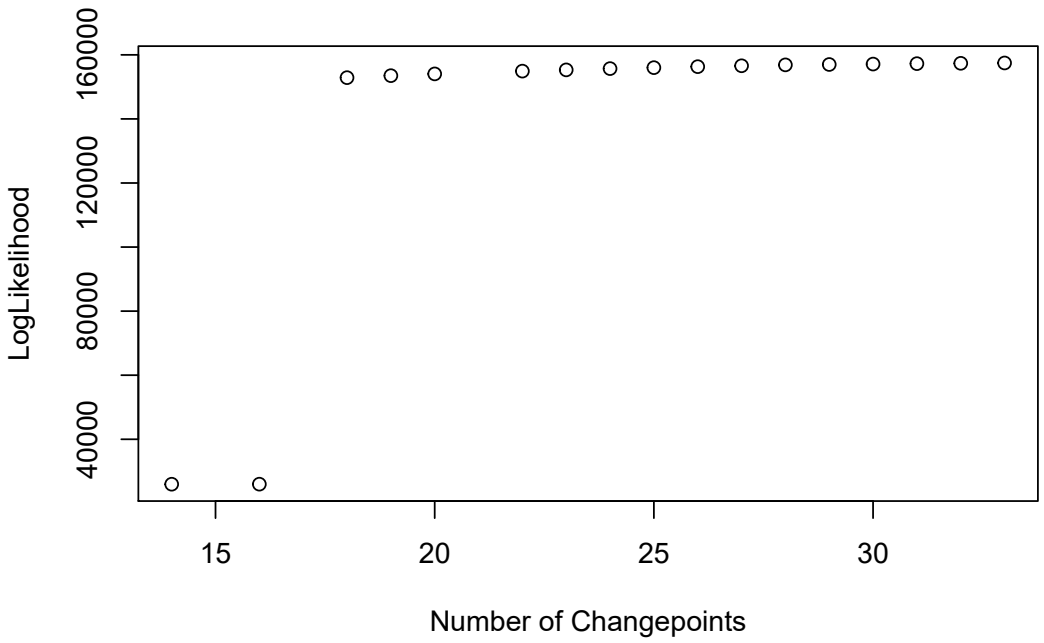

Figure 4.3: CROPS scree plot for biweight loss minimization

The scree plot for the chosen range of penalties is shown in Figure 4.3. There is a huge leap in the likelihood, probably meaning that for the maximum penalty in the range the model is underfit. Since the penalties in the range that give change-point models with minimum segment length less than 30 are discarded due to overfitting, the optimal value can be surely found in the scree plot.

This huge leap makes it difficult to see where the optimal change-point model is among the penalties chosen by CROPS, so the first two values will be cut from the plot

```
plot(crops.biw$ncp[-(1:2)],cpm.loss.biw.ll[-(1:2)]
     ,xlab="Number of Changepoints"
     ,ylab="LogLikelihood")
```

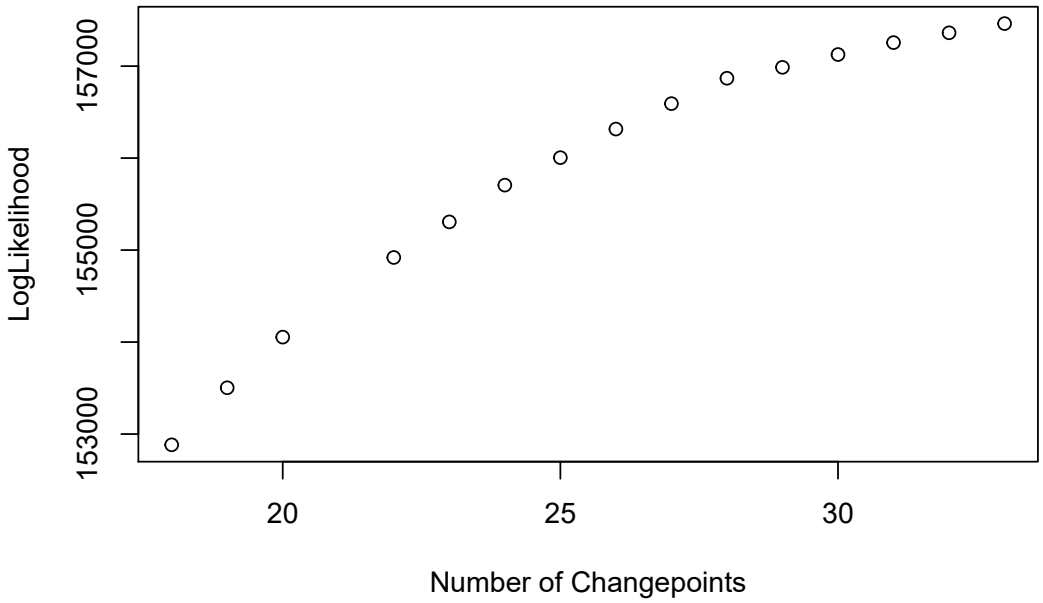

Figure 4.4: CROPS zoomed scree plot for biweight loss minimization

A zoom of the scree plot has been made and an "elbow" is not so evident, though a choice of 28 change-points may represent a good compromise between a high value of the approximated likelihood and a low risk of finding a segment which is too short. The penalty to be used is retrieved from the row of the dataframe `crops.biw`.

```
cpm.loss.ll.opt<-cpm.loss.biw.ll[which(crops.biw$ncp==28)]
beta.sel=crops.biw$beta[which(crops.biw$ncp==28)]
cbind(crops.biw[which(crops.biw$ncp==28),-3],logl=cpm.loss.ll.opt)
```

```
##    ncp     beta beta.log msl     logl
## 12  28 46.82276 4.225939  58 156868.3
```

The model is then saved and, in Figure 4.5 plotted along with the estimated change-point locations (blue vertical lines) and the M-estimate of the mean for

each segment (red lines).

```
cpm.loss.biw<-Rob_seg.std(s_,loss="Outlier",lambda=beta.sel,lthreshold=3)
smt=cpm.loss.biw$smt*sigmah
plot(s,ylim=c(25,27),type="l")
lines(smt,col="red")
abline(v=cpm.loss.biw$t.est[-cpm.loss.biw$K],col="blue")
```

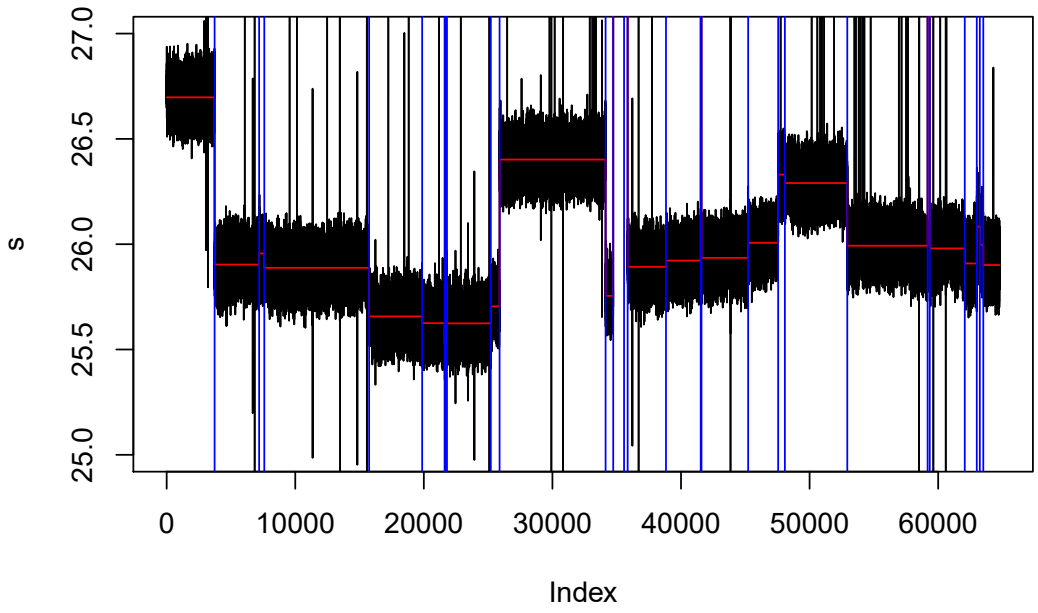

Figure 4.5: Optimal change-point model based on biweight loss

The model seems reasonable: by looking at the plot the change-points are not too many, neither too close to each other and the visible changes are detected; also the presence of outliers seem not to affect at all the model.

4.3 Binary Segmentation

A popular class of change-point algorithms is represented by the binary segmentation, first introduced by Vostrikova [19].

Given a segment $\boldsymbol{Y}$ with length N its binary segmentation consists in three phases:

1. find a change-point candidate $\hat{\tau}$ among all the points in $\boldsymbol{Y}$

2. test if a change-point is actually present in $\hat{\tau}$
3. if a change-point is present in $\hat{\tau}$ perform binary segmentation on $\boldsymbol{Y}_{(1:\hat{\tau}-1)}$ and $\boldsymbol{Y}_{(\hat{\tau}:N)}$

Ideally phase 1 is performed by using a measure $A_i(\boldsymbol{Y})$ which tells, for each point i, how likely is a structural break at i and the point where this number is maximized will represent a change-point candidate.

So, at every iteration of the algorithm

$$\hat{\tau} = \arg\max_i A_i(\boldsymbol{Y})$$

In this sense, the binary segmentation is a greedy algorithm since, according to measure $A_i(\boldsymbol{Y})$, at every iteration the optimal choice for one possible change-point is made.

Once this is done, the candidate change-point has to be tested. It is common to do the hypothesis test using:

$$H_0 : \text{No change-point is present}, \quad H_A : \text{A change-point in } \hat{\tau} \text{ is present}$$

and a measure $A_i(\boldsymbol{Y})$ so that, under H_0, $M_A = \max_i A_i(\boldsymbol{Y})$ has a cumulative distribution $F_M(x)$. This will be the case of the two statistics which will be used in a binary segmentation setting in order to detect changes in mean on the manufacturing data.

4.3.1 CUSUM statistic

The CUSUM is a statistic initially proposed by Page in 1954 for statistical process control purposes. Montgomery [20] describes it as a valid alternative to the Stewhart control chart, more powerful in detecting small ($\leq 1.5\sigma$) structural breaks.

Definition 4.3.1. *Given a time series Y_t and an estimate of the location parameter $\hat{\mu}_Y$ the CUSUM statistic for the i-th sample is built as it follows*

$$C_i = \sum_{j=1}^{i} (Y_j - \hat{\mu}_Y)$$

Where $\hat{\mu}_Y = \frac{1}{N}\sum_{j=1}^{N} Y_j$. The CUSUM statistic has become very popular in statistical control since it is easy and fast to calculate.

In order to detect a structural break, the CUSUM statistic, given the data has to be tested. the hypotheses will be:

$$H_0 : \mu_i = \mu \quad \forall i$$

Meaning that there is no evidence for a change-point in mean, and

$$H_A : \exists \tau : \mu_\tau \neq \mu_{\tau+1}$$

Meaning that a change-point is present. The point where the absolute value of the CUSUM statistic is the highest will be considered as the possible change-point: as example, if a positive change in mean is present the points before it are likely to be below $\hat{\mu}$, making the CUSUM decrease, whereas after the change-point the CUSUM will increase, as the points will likely be above $\hat{\mu}$, thus the maximum of the CUSUM absolute value is likely to be the change-point.

The CUSUM statistic for independent variables has another advantage: under H_0 its asymptotic behaviour can be obtained using the so-called Functional Central Limit Theorem, which has been proven by Donsker [21].

Theorem 4.3.1 (Donsker)**.** *Let $\boldsymbol{X} = \{X_i\}$ be a set of N i.i.d. random variables so that $E[X] = 0$ and $Var[X] = 1$. Let $S_i = \sum_{j=1}^{i} X_i$ be the set of the cumulative sums and let $W_t^{(N)} = \frac{S_{\lfloor Nt \rfloor}}{\sqrt{N}}$ be the partial-sum process rescaled to $t \in [0,1]$. Then*

$$W_t^{(N)} \to W(t) \qquad t \in [0,1]$$

is convergent in distribution as $N \to \infty$, where $W(t)$ is the standard brownian motion.

By Definition 4.3.1, the CUSUM statistic can be rewritten as the difference of two sums

$$C_i(\boldsymbol{Y}) = \sum_{j=1}^{i} Y_j - \frac{i}{N} \sum_{j=1}^{N} Y_j$$

If Y_j are i.i.d. random variables with mean μ and variance σ^2, in order to apply Theorem 4.3.1 it can be observed that

$$\frac{C_i(\boldsymbol{Y})}{\hat{\sigma}} = \sum_{j=1}^{i} \frac{(Y_j - \mu)}{\hat{\sigma}} - \frac{i}{N} \sum_{j=1}^{N} \frac{(Y_j - \mu)}{\hat{\sigma}}$$

where σ is a (asymptotically) consistent estimator of the standard deviation, is composed of two sums of random variables with mean 0 and variance 1.

Now the process can be rescaled, this means that $i = \lfloor Nt \rfloor$

$$\frac{C_{\lfloor Nt \rfloor}(\boldsymbol{Y})}{\hat{\sigma}\sqrt{N}} \to W(t) - tW(1) = B(t)_{[0,1]} \tag{4.5}$$

in distribution as $N \to \infty$. The process $B(t)_{[0,1]}$ is called a Brownian Bridge defined over the interval [0,1].

Under H_0 with large samples the variable to be tested follows the same distribution of $\max |B(t)_{[0,1]}|$ which, as proven by Shorack and Wellner [22] follows a Kolmogorov Distribution

$$P\left[\frac{1}{\sqrt{N}\hat{\sigma}} \max_i |C_i(\boldsymbol{Y})| < x\right] = \frac{2\pi}{x} \sum_{j=1}^{\infty} \exp\left(-\frac{(2j-1)^2\pi^2}{8x^2}\right) \tag{4.6}$$

Given this result it is possible to test the hypothesis with significance α for a single change-point.

Robust CUSUM

The presence of extreme outliers may have a negative impact on the CUSUM statistic: suppose that an observation $\omega \in \boldsymbol{Y}$ at time t_ω can be arbitrarily large, then

$$\exists \omega^* : \omega > \omega^* \Longrightarrow \text{argmax}_t C_t = t_\omega$$

meaning that, even if a change-point is present in a different location, the CUSUM statistic will prefer t_ω.

Also, in case of i.i.d. variables (no structural breaks)

$$\exists \omega^* : \omega > \omega^* \Longrightarrow \frac{1}{\sqrt{N}\hat{\sigma}} \max_i |C_i(\boldsymbol{Y})| > K_{1-\alpha} \quad \forall \alpha \in (0,1)$$

where $K_{1-\alpha}$ is the $1-\alpha$ quantile of the Kolmogorov distribution, this means that a structural break will be detected in presence of a sufficiently large outlier.

It can be stated that even the presence of a single outlier is enough to give inaccurate results when the CUSUM statistic is used. In order to overcome this issue an alternative using a biweight-like transformation is presented.

The main idea is that, as in the loss-minimizing functions, a CUSUM-like measure on the set $\boldsymbol{Y}$ can be obtained by using the some data transformation $\phi_K(\boldsymbol{Y})$ which has to be particularly robust to extreme outliers.

$$\phi_K(\boldsymbol{Y}) = \begin{cases} K & Y_i > \hat{\mu} + K \\ -K & Y_i < \hat{\mu} - K \\ Y_i - \hat{\mu} & \text{otherwise} \end{cases}$$

where $\hat{\mu}$ is the M-estimator of the location parameter using equation 2.3. This transformation is similar to the Biweight Loss applied in section 4.2 and it can be seen that, even in presence of an extreme outlier ω at time t_ω, the robust CUSUM statistic

$$C_i\left(\phi_K(\boldsymbol{Y})\right)$$

treats it as if it were $\hat{\mu} \pm K$ (depending on the value of ω) . Note that, in order to obtain a sensible CUSUM statistic, K has to be chosen properly, this involves the use of a robust estimator of the scale parameter as in 2.3.

A binary segmentation algorithm based on robust CUSUM is implemented in the function `binseg.rob.cusum`, a slightly modified version of `binseg.mean.cusum` from the package `changepoint` by Killick [23] [24]. The original function from `changepoint` has several input parameters: `Q` is the maximum number of change-points, after which the algorithm stops; the parameter `pen` is set at the 95% of the Kolmogorov cumulative distribution (for which package `CPAT` [25] is used), this is equivalent to say that, at every iteration, a test with $\alpha = 5\%$ is performed in order to verify if a change-point is present; the minimum segment length `minseglen` is set to 30 in order to avoid the segmentation of groups of outliers.

```
cpm.bs.cus.95<-binseg.rob.cusum(s,Q=200,
pen=CPAT:::pkolmogorov(0.95),minseglen = 30)
cpm.bs.cus.95$op.cpts
```

```
## [1] 112
```

The number of change-points found is much higher than what found in section 4.2, hence there is a risk of overfitting. It is necessary, as in the robust loss minimization, to tune the penalty parameter in order to avoid it and the approximated likelihood function from Chapter 3 will help.

For each step of the binary segmentation (at which a change-point is added), the approximated likelihood, given the change-points, is maximized and collected in the vector `cpm.bs.cus.ll` with the aim of capturing the evolution of the approximated likelihood through a scree plot.

```
plot(0:cpm.bs.cus.95$op.cpts,cpm.bs.cus.ll,
xlab="# of change-points",ylab="log-likelihood")
```

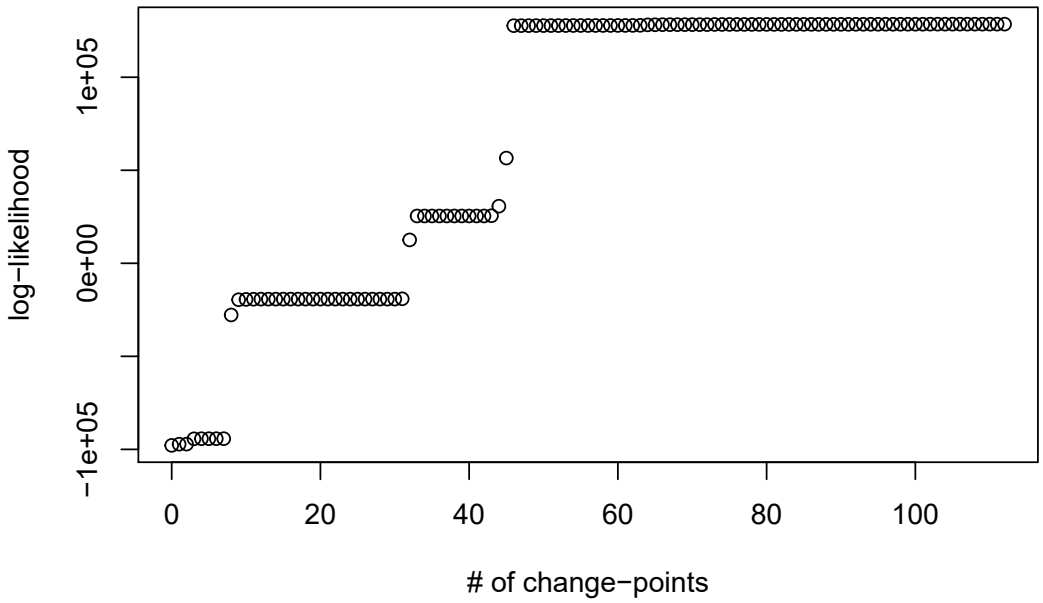

Figure 4.6: Evolution of the approximated log-likelihood in binary segmentation using CUSUM

The scree plot in Figure 4.6 shows an almost step-wise evolution of the approximated log-likelihood, with over 40 iterations necessary to reach a high value.

The optimal segmentation can be found once the high value is reached. Since in figure 4.6, starting from 46 changepoints, the differences in the log-likelihood cannot be seen, a zoomed plot is presented.

```
plot(46:cpm.bs.cus.95$op.cpts,cpm.bs.cus.ll[47:length(cpm.bs.cus.ll)],
xlab="# of change-points",ylab="log-likelihood")
```

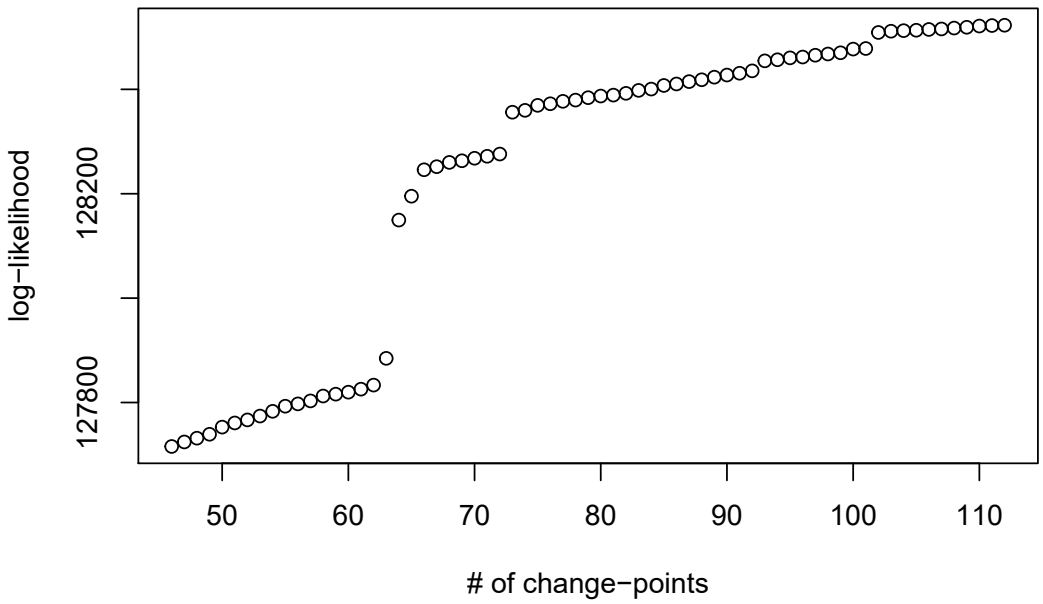

Figure 4.7: Evolution of the approximated log-likelihood in binary segmentation using CUSUM (zoom)

Figure 4.7 shows how the likelihood grows after the last "step", which is the most interesting part of the plot since the optimal segmentation will be found there. By looking at the plot a sort of "elbow" can be found at 66 change-points: this will be the optimal model.

```
cpm.cus.ll.opt<-cpm.bs.cus.ll[67]
cbind(65:67,t(cpm.bs.cus.95$cps)[65:67,])
```

```
##      [,1]  [,2]     [,3]
## [1,]   65 21492 1.148052
## [2,]   66 21613 1.148052
## [3,]   67   933 1.132076
```

The penalty is chosen according to the number of change-points at the elbow, then the optimal model is plotted in Figure 4.8, with the change-points represented by the blue vertical lines.

```
cp.cus.pen<-1.14
cpm.bs.cus<-binseg.rob.cusum(s,adj=F,Q=67,pen=1.14,minseglen = 30)
```

```
plot(s,ylim=c(25,27),type="l")
abline(v=cpm.bs.cus$cpts,col="blue")
```

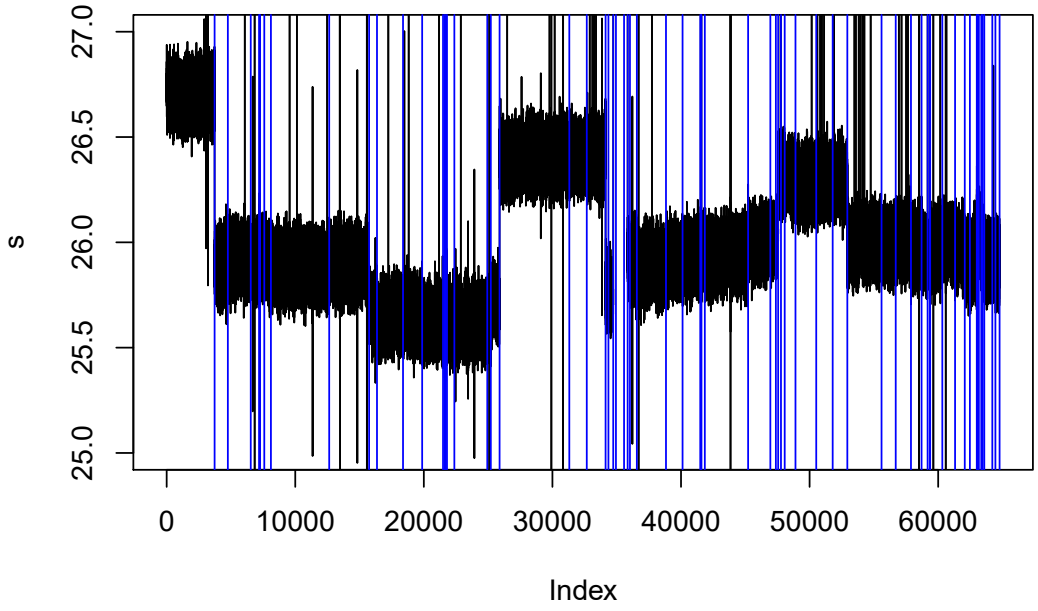

Figure 4.8: Optimal change-point model based on binary segmentation using CUSUM statistic

The application of the binary segmentation using a robust CUSUM statistic shows actual robustness to outliers along with the fact that the segments are well separated even though there is some tendency to overfitting.

The CUSUM statistic as it is has a major downside: if an actual structural break is present towards the extremities of the segment, the estimate of the change-point through its maximum may be inaccurate, since it can present a non-negligible bias and a large variance. In order to put this effect into evidence, 10^4 simulations (for each change-point) of $N = 10^4$ standard normal random variables are performed:

a single change-point in mean is present in different locations (t.true) and the difference between the two means is set equal to σ. Sample mean(mean.t) and sample variance(var.t) of the change-point estimate $\hat{\tau}$ are collected. The value of bias.t is calculated as the difference between the true value and the sample mean.

t.true	mean.t	bias.t	var.t
5000	5000.048	0.0480	5.3414
7000	6997.956	-2.0436	19.3738
8000	7994.821	-5.1793	97.9985
9000	8978.822	-21.1783	1365.4576
9500	9414.392	-85.6079	20608.5046

Table 4.1: Simulation results for CUSUM statistics

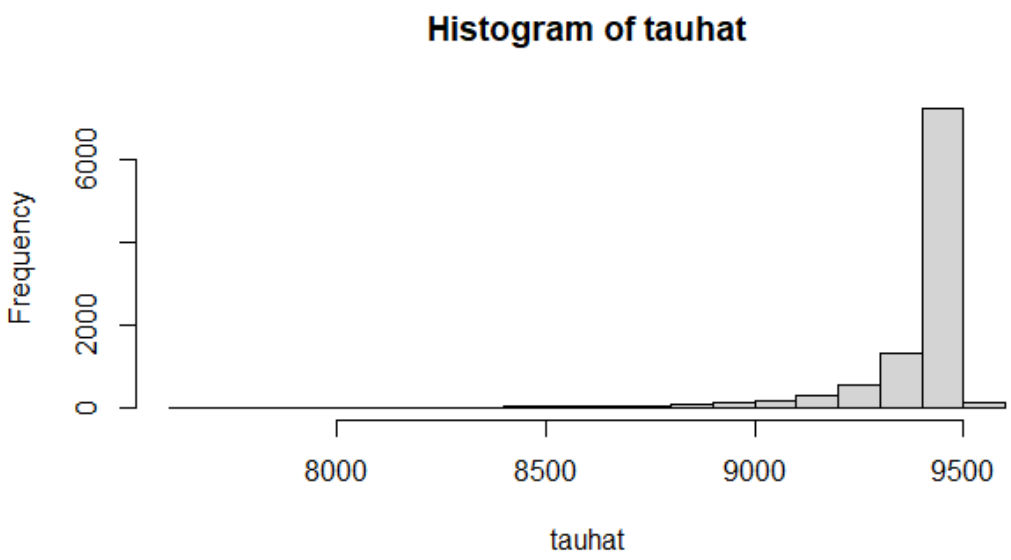

Figure 4.9: Distribution of $\hat{\tau}$ using CUSUM

In particular when τ equals 9000 or 9500 the estimate $\hat{\tau}$ is biased towards the center of the segment.

The histogram of the estimates $\hat{\tau}$ for $\tau = 9500$ is reported in figure 4.9, a similarity with a truncated exponential can be noticed.

In order to overcome this issue, especially when the segmentation is not symmetric (which can be quite common in industrial applications), an adjusted version of the

CUSUM statistic is presented.

4.3.2 adjusted CUSUM statistic

Definition 4.3.2. *Given a time series Y_t and an estimate of the location parameter $\hat{\mu}_Y$ the adjusted CUSUM statistic $\tilde{C}_i$ for the i-th sample is defined as it follows*

$$\tilde{C}_i = \frac{C_i(Y_t)}{\sqrt{\frac{i}{N}\left(1 - \frac{i}{N}\right)}}$$

where $C_i(Y_t)$ is the CUSUM statistic as in Definition 4.3.1

The squared version of this statistic is well documented by Robbins [26]. This weighted version of the CUSUM can deal better with change-points located at the extremities of the segment, it can be noted that the weights depend on i and the minimum weight is at $i = \frac{N}{2}$.

The same simulation as in table 4.1 is made for the adjusted CUSUM, in order to see how biases and variances of the change-point location change when the statistic is adjusted.

t.true	mean.t	bias.t	var.t
5000	5000.0473	-0.0473	5.3464
6000	6000.0098	-0.0098	5.0510
7000	6999.987	0.0130	4.9313
8000	7999.9994	0.0006	4.8033
9000	8999.9886	0.0114	4.7459
9500	9499.9867	0.0133	5.2302

Table 4.2: Simulation results for CUSUM-adj statistics

The simulations made with the adjusted CUSUM statistics are summed up in table 4.2, the most interesting results is that bias and variance do not significantly change with the position of the change-point, mind that this result does not hold for the normal CUSUM.

The histogram of $\hat{\tau}$ obtained using the adjusted CUSUM is reported in figure 4.10, unlike CUSUM, the estimate of the change-point presents a bell curve centered around the true change-point value, the range of values is also smaller (which is expectable due to the much lower variance).

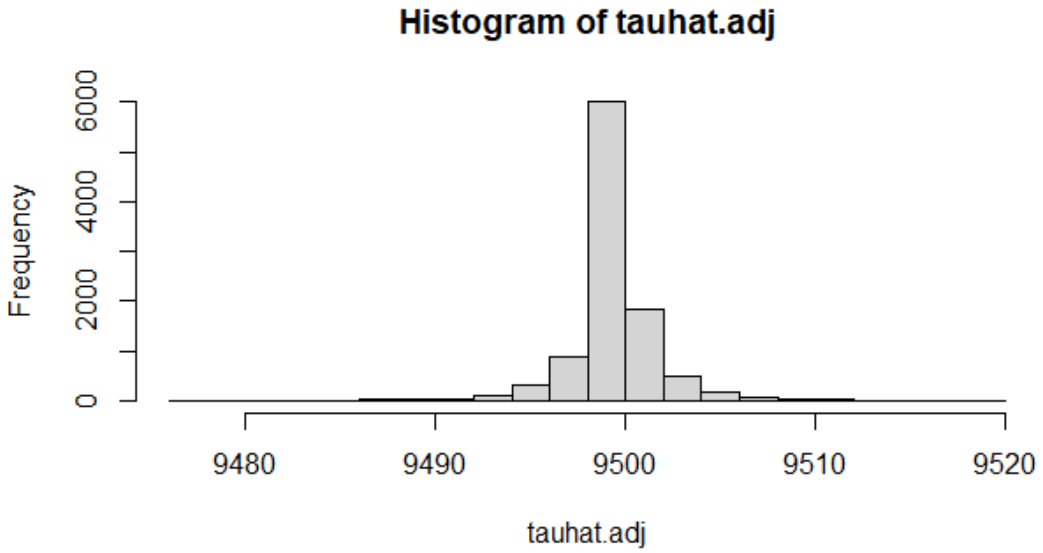

Figure 4.10: Distribution of $\hat{\tau}$ using adjusted CUSUM

The asymptotic behaviour of the square of the adjusted-CUSUM statistic is well explained by Robbins [26], who states and proves the following theorem using the Reflection Principle.

Theorem 4.3.2. *Let $\boldsymbol{Y}$ be a set of N i.i.d. random variables, let $\tilde{C}_i(\boldsymbol{Y})$ be its adjusted CUSUM statistic and let $\hat{\sigma}^2$ be a consistent estimator of the variance Then the following convergence in distribution holds:*

$$\max_{Nl \leq i \leq Nh} \frac{\tilde{C}_i^2(\boldsymbol{Y})}{N\hat{\sigma}^2} \rightarrow \max_{t \in [l,h]} \left(\frac{B^2(t)}{t(1-t)} \right) \quad \forall \{l,h\} : 0 < l < h < 1$$

Where $B^2(t)$ denotes the squared brownian bridge over [0,1].

Although the exact asymptotic distribution of $\max_{t \in [l,h]} \left(\frac{B^2(t)}{t(1-t)} \right)$ is not known, Csörgö and Horváth in 1993 [27] came up with an approximation of its tail probability, also reported in [26], which is particularly useful in order to build a statistical test for the adjusted CUSUM.

$$P\left[\sup_{t \in [l,h]} \left(\frac{B^2(t)}{t(1-t)} \right) > x \right] \rightarrow \sqrt{\frac{xe^{-x}}{2\pi}} \left[1 + \frac{3}{x} + \log\left(\frac{h(1-l)}{(1-h)l} \right) + \mathcal{O}(x^{-2}) \right] \quad (4.7)$$

This result is implemented in the function `binseg.rob.cusum`. If the parameter `adj` is set to true the parameter `pen` becomes the α of the hypothesis test, in which the values of l and h will be obtained by setting l as the ratio between the total segment length and the parameter `minseglen`. h will be set $1 - l$.

The first model using adjusted CUSUM is applied with $\alpha = 5\%$ at each segmentation.

```
cpm.bs.adj.95<-binseg.rob.cusum(s,Q=50,pen=0.05,
                                adj=T,minseglen = 30)
```

```
cpt.bs.adj.95<-cpm.bs.adj.95$cpts
cpm.bs.adj.95$op.cpts
```

```
## [1] 43
```

The number of change-points found is much lower than the segmentation using standard CUSUM. The use of this adjusted CUSUM statistic in change-point analysis presents an improvement against overfitting.

As made for the previous model, the log-likelihood for all segmentations is maximized and stored in the variable `cpm.bs.adj.ll`, in order to draw a scree plot (Figure 4.11) that can be helpful for finding the optimal model.

The code below draws a plot of the evolution of the approximated log-likelihood as the segmentation goes on until 50 change-points. The segmentation performed with penalty $\alpha = 5\%$ is highlighted with a black vertical line.

```
plot(0:50,cpm.bs.adj.ll,xlab="# of change-points",ylab="log-likelihood")
abline(v=cpm.bs.adj.95$op.cpts)
```

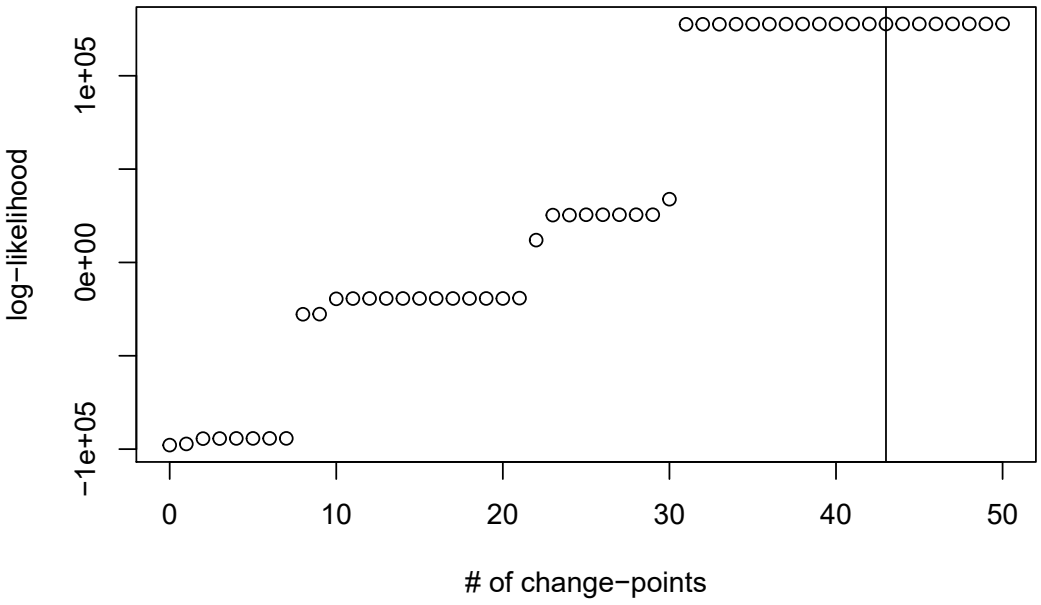

Figure 4.11: Evolution of the approximated log-likelihood in binary segmentation using adjusted CUSUM (black vertical line is the segmentation with penalty =5%)

The plot confirms a better resistance to overfitting of the adjusted CUSUM with respect to the standard CUSUM, since the approximated likelihood takes the last step once 31 change-points are found. As in the previous model, it can be of interest to inspect the plot starting from (in this case) 31 change-points.

A zoomed version of the plot is computed in order to see how the log-likelihood evolves once a good value is reached.

```
plot(31:50,cpm.bs.adj.ll[32:length(cpm.bs.adj.ll)],
xlab="# of change-points",ylab="log-likelihood")
```

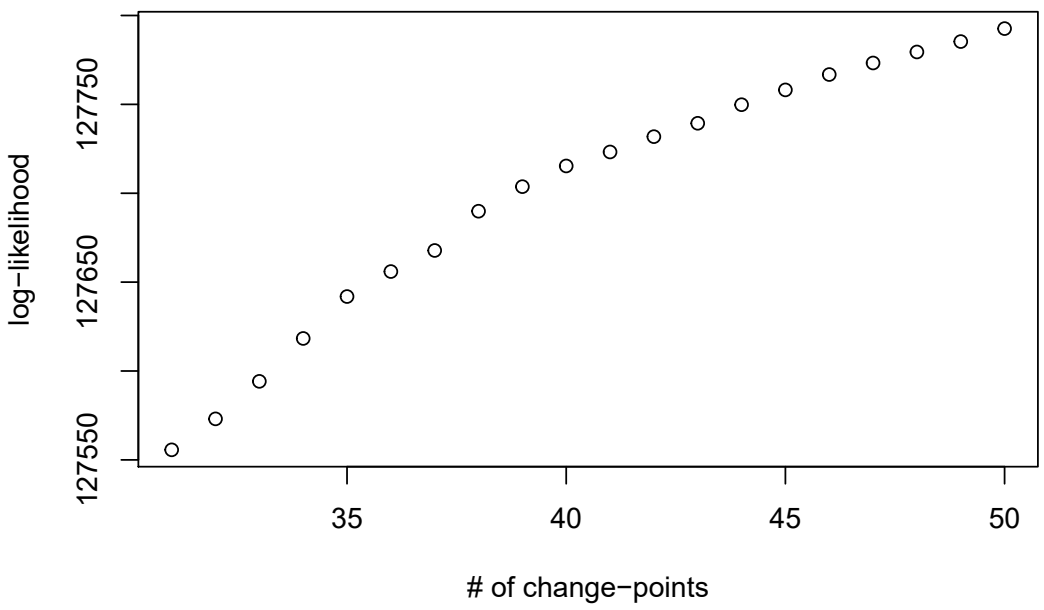

Figure 4.12: Evolution of the approximated log-likelihood in binary segmentation using adjusted CUSUM (zoom)

By looking at the zoomed plot it is hard to see an elbow, however the choice of 38 plots might be a good compromise between a good value of the likelihood and the low risk of overfitting.

Once the choice for the number of change-points is made the penalty can be chosen.

```
cpm.adj.ll.opt<-cpm.bs.adj.ll[39]
cbind(37:39,t(cpm.bs.adj.95$cps)[37:39,])
```

```
##      [,1]  [,2]         [,3]
## [1,]   37 48898 -0.003496199
## [2,]   38 50511 -0.003496199
## [3,]   39  7225 -0.005617719
```

The optimal model is nw applied and plotted.

```
cpm.bs.adj<-binseg.rob.cusum(s,adj=T,Q=39,pen=5e-3,minseglen = 30)
```

```
plot(s,ylim=c(25,27),type="l")
abline(v=cpm.bs.adj$cpts,col="blue")
```

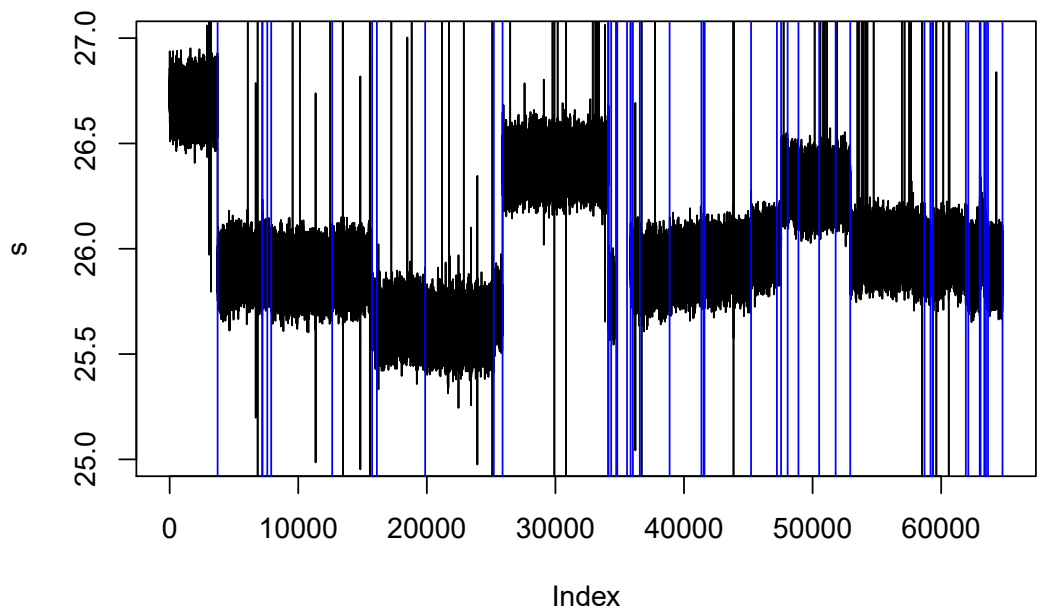

Figure 4.13: Optimal change-point model based on binary segmentation using adjusted CUSUM statistic

The plot shows that some improvements regarding the overfitting problems are

made with respect to what seen in Figure 4.8, as well as the precision of change-point detection (as shown before). With these premises the adjusted CUSUM model might be preferrable to the standard CUSUM.

4.4 Model selection

Three robust models for change-point detection are explained and applied to the dataset. The following dataframe compares the approximated loglikelihood of the three models and the number of change-points found.

```
data.frame(model=c("Loss Function","Standard CUSUM","Adjusted CUSUM"),
loglik=c(cpm.loss.ll.opt,
cpm.bs.ll.opt,
cpm.adj.ll.opt),ncpts=c(28,66,38))
```

```
##            model   loglik ncpts
## 1  Loss Function 156868.3    28
## 2 Standard CUSUM 128245.9    66
## 3 Adjusted CUSUM 127689.9    38
```

There is no need to use information criteria in order to select the best model, since the one based on robust loss function minimizing has by far the greatest approximated likelihood and the lowest number of change-points found, which means essentially the lowest number of parameters, among the models.

Now that the model is chosen, the maximum approximated likelihood parameter estimates are presented.

```
model.approx.par
```

```
## $mu
##  [1] 26.69734 25.90351 25.95661 25.88772 25.65715 25.62552
##  [7] 25.52690 25.81984 25.62431 25.70532 26.40200 25.75512
## [13] 56.96396 45.95599 25.89241 25.92240 44.93625 25.93543
## [19] 26.00642 26.33074 26.29060 25.99285 46.00586 25.97972
## [25] 25.90922 26.08304 25.99866 25.90093
##
## $sigma2
## [1] 0.001263386
##
## $sigma2e
## [1] 0.003066674
##
```

```
## $cpts
##  [1]  3728  7190  7592 15744 19870 21613 21713 21800
##  [9] 25206 25890 34135 34729 35572 35850 38832 41533
## [17] 41591 45222 47556 48076 52932 59179 59345 62067
## [25] 63002 63239 63508 64776
```

By looking at the variances it can be observed that σ_ε^2 is estimated to be larger than σ^2, which is consistent with the fact that the robust autocorrelation at lag 1 (Figure 1.11) is close to 0.5.

Another thing to be noticed is that a group of consecutive means close to each other is present (as an example means n. 5 and 6), however, due to the low variance a test hypothesis of means equality may be rejected. Tests or confidence intervals based on this approximate model have no point to be built, since a more accurate model will be developed in the next chapter.

Chapter 5

Bayesian Model with measurement delays

This chapter will be dedicated to the development of a more accurate model for the manufacturing process analyzed in the previous chapters. The new model will have to overcome the normal approximations used in chapter 3, since by definition an inter-event time in a process cannot be negative, neither a delay in measurement.

5.1 The Hierarchical Model

The construction of the following model assumes that the vector of change-points $\boldsymbol{\tau}$ is known. In chapter 4 the vector $\boldsymbol{\tau}^c = \{0, \boldsymbol{\tau}, T\}$ is defined starting from it.

The aim is to write the a-posteriori probability as

$$p(\boldsymbol{\theta}, \boldsymbol{\varepsilon}|\boldsymbol{Y}) \propto p(\boldsymbol{Y}, \boldsymbol{\varepsilon}, \boldsymbol{\theta}) = p(\boldsymbol{Y}|\boldsymbol{\varepsilon}, \boldsymbol{\theta}) p(\boldsymbol{\varepsilon}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{5.1}$$

Since all elements in $\boldsymbol{X}$ and $\boldsymbol{\varepsilon}$ have to be positive, it is possible to model them as lognormal random variables.

This means that

$$\log(X_i) \sim \mathcal{N}(m_j, s_j^2) \iff \tau_{j-1}^c < i \leq \tau_j^c$$

$$\log(\varepsilon) \sim \mathcal{N}(m_\varepsilon, s_\varepsilon^2)$$

Reparametrizations

Note that, even if the change-points in the expected value are present both parameters of the lognormal will be affected by it.

This may be an issue, since there is no proof that the variance of the elements in $\boldsymbol{X}$ changes when a change-point in mean is detected.

This means that a reparametrization of $\{m_j, s_j^2\}$ is necessary, in order to describe the fact that any change-point in the process is just a change of the mean cycle time.

From the properties of the log-normal distribution, if X_i belongs to the segment j then

$$\mu_j = E[X_i] = \exp\left(m_j + \frac{s_j^2}{2}\right) \qquad \text{and} \qquad \sigma_j^2 = Var[X_i] = \exp(2m_j + s_j^2)(e^{s_j^2} - 1)$$

by performing a simple inversion it is possible to obtain the parameters of the lognormal as functions of $\{\mu_j, \sigma_j^2\}$

$$m_j(\mu_j, \sigma_j^2) = 2\log\mu_j - \frac{1}{2}\log(\mu_j^2 + \sigma_j^2) \qquad s_j^2(\mu_j, \sigma_j^2) = \log\left(1 + \frac{\sigma_j^2}{\mu_j^2}\right) \tag{5.2}$$

With the new parametrization and the hypothesis that $\sigma_j^2 = \sigma^2$ for all j, it is possible to write

$$\log(Y_i - \varepsilon_i + \varepsilon_{i-1})|\boldsymbol{\varepsilon} \sim \mathcal{N}\left(m_j(\mu_j, \sigma^2), s_j^2(\mu_j, \sigma^2)\right) \iff \tau_{j-1}^c < i \leq \tau_j^c \tag{5.3}$$

meaning that $Y_i|\boldsymbol{\varepsilon}$ follows a log-normal distribution shifted by the quantity $\varepsilon_i - \varepsilon_{i-1}$ with two parameters, both depending on the mean(which depends on the segment where i is found) and the variance, which is the same for all segments.

If $\boldsymbol{X}$ is assumed as a set of independent random variables , then the variables in $\boldsymbol{Y}|\boldsymbol{\varepsilon}$ are independent, it follows that

$$p(\boldsymbol{Y}|\boldsymbol{\varepsilon}, \boldsymbol{\theta}) = \prod_i p(Y_i|\boldsymbol{\varepsilon}, \boldsymbol{\theta})$$

There is, as discussed in the approximated model, an identifiability problem with the expected value of ε, since only the difference between two consecutive measurement delays can be observed as part of the process. In order to overcome the problem the distribution of any ε_i is set to have a fixed expected value μ_ε. Mind that setting a fixed value for μ_ε does not mean that there is a belief that $E[\varepsilon]$ is equal to that fixed value.

In this setting the mean of the logarithm becomes a function of the variance of the logarithm.

$$m_\varepsilon = \log(\mu_\varepsilon) - \frac{1}{2}s_\varepsilon^2$$

Since the variables in $\boldsymbol{\varepsilon}$ are assumed independent the a posteriori probability distribution becomes proportional to

$$p(\boldsymbol{Y}, \boldsymbol{\varepsilon}, \boldsymbol{\theta}) = \prod_{i=1}^{N} p(Y_i|\boldsymbol{\varepsilon}, \boldsymbol{\theta}) \prod_{i=0}^{N} p(\varepsilon_i|\boldsymbol{\theta}) \prod_{i=1}^{N_p} p(\theta_i) \tag{5.4}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \sigma^2, s_\varepsilon^2\}$ is the vector of parameters and $N_p = |\boldsymbol{\theta}|$.

The priors for the parameters are chosen as non-informative on a positive support: for σ^2 and s_ε^2 inverse gammas are chosen, whereas the priors for $\mu_i \in \boldsymbol{\mu}$ are chosen as lognormals, since in an injection moulding process the order of magnitude of the cycle time may vary from $\mathcal{O}(1s)$ to $\mathcal{O}(100s)$ (see i.e. chapter 7 of [28]) .

$$\log(\mu_i) \sim \mathcal{N}(3{,}4) \qquad \forall \mu_i \in \boldsymbol{\mu}$$

$$t_X = \frac{1}{\sigma^2} \sim \Gamma(10^{-3}{,}10^{-3})$$

$$t_\varepsilon = \frac{1}{\sigma^2} \sim \Gamma(10^{-3}{,}10^{-3})$$

5.2 Metropolis-Within-Gibbs

The Gibbs sampling cannot be performed due to the fact that the full conditional distributions cannot be resembled to a known kernel. In this case a variation of the Gibbs sampling based on the Metropolis algorithm [29] can be applied.

The algorithm starts with a vector of initial values $\boldsymbol{\varphi}^{(0)}$, one for each variable in $\boldsymbol{\varphi} = \{\boldsymbol{\varepsilon}, \boldsymbol{\theta}\}$. A candidate for the first parameter φ_1^* is chosen by sampling from a normal distribution with mean $\varphi_1^{(0)}$, then an acceptance rate ν is calculated as

$$\nu = \frac{p(\varphi_1^* | \boldsymbol{\varphi}_{-1}^{(0)})}{p(\varphi_1^{(0)} | \boldsymbol{\varphi}_{-1}^{(0)})}$$

which is the ratio between the full conditional probability functions valued in the candidate and in the old value. Now a random number from Uniform(0,1) is sampled, if it is lower than ν, then $\varphi_1^{(1)} = \varphi_1^*$, else $\varphi_1^{(1)} = \varphi_1^{(0)}$.

The procedure is repeated for all parameters, where the acceptance rate is, for a generic φ_i, calculated as

$$\nu = \frac{p(\varphi_i^* | \boldsymbol{\varphi}_{(1:i-1)}^{(1)}, \boldsymbol{\varphi}_{(i+1:M)}^{(0)})}{p(\varphi_i^{(0)} | \boldsymbol{\varphi}_{(1:i-1)}^{(1)}, \boldsymbol{\varphi}_{(i+1:M)}^{(0)})}$$

conditioned on the updated parameters. These steps are repeated for all M parameters, for a sufficient number K of iterations. The result is a Markov chain by which, if converges, it is possible to extract the a posteriori distributions of the parameters.

There is no need to calculate the full conditionals every time that a new parameter is sampled and evaluated: many times these distributions can be set as proportional to a much simpler probability function.

It can be seen that a change in ε_0 changes only two terms of the product

$$p(\varepsilon_0 | \boldsymbol{Y}, \boldsymbol{\varepsilon}_{(-0)}, \boldsymbol{\theta}) \propto p(Y_1 | \boldsymbol{\varepsilon}, \boldsymbol{\theta}) p(\varepsilon_0 | \boldsymbol{\theta}) \tag{5.5}$$

The same can be said for ε_N

$$p(\varepsilon_N | \boldsymbol{Y}, \boldsymbol{\varepsilon}_{(-N)}, \boldsymbol{\theta}) \propto p(Y_N | \boldsymbol{\varepsilon}, \boldsymbol{\theta}) p(\varepsilon_0 | \boldsymbol{\theta}) \tag{5.6}$$

While a generic ε_i in the middle of the process changes three terms, since it affects cycle times i and $i+1$

$$p(\varepsilon_i | \boldsymbol{Y}, \boldsymbol{\varepsilon}_{(-i)}, \boldsymbol{\theta}) \propto p(Y_i | \boldsymbol{\varepsilon}, \boldsymbol{\theta}) p(Y_{i+1} | \boldsymbol{\varepsilon}, \boldsymbol{\theta}) p(\varepsilon_i | \boldsymbol{\theta}) \quad \forall i \in 1 \dots N-1 \tag{5.7}$$

It can be also observed that a change in μ_j affects only the cycle times belonging to segment j, as well as its prior.

$$p(\mu_j | \boldsymbol{Y}, \boldsymbol{\varepsilon}, \boldsymbol{\theta} \setminus \mu_j) \propto p(\mu_j) \prod_{i=\tau_j^c+1}^{\tau_j^c} p(Y_i | \boldsymbol{\varepsilon}, \boldsymbol{\theta}) \tag{5.8}$$

5.3 Application and Results

Before fitting the model to the data some considerations on both the data and the model have to be made:

- Outliers are treated as missing data: the imputation (substitution of the outliers with "normal" data) is made so that the approximated likelihood seen in 3.3 is maximized. This imputation is done in order to avoid outliers to influence the a-posteriori distribution
- Different sets of initial values are proposed in input, in order to check if the Markov chain converges always to the same distribution.
- The proposal standard deviations are chosen by inspection, so that the acceptance rate is sufficiently close to the value 0.234 proposed by Roberts Gelman and Gilks [30]. In this case all $\varepsilon_i \in \boldsymbol{\varepsilon}$ have equal proposal distribution and for any $\mu_i \in \boldsymbol{\mu}$ the proposal standard deviation is proportional to the square root of the respective segment length. Another parameter that has to be set is μ_ε which is arbitrarily chosen to be equal to 1.
- At each iteration the logarithm of the posteriori is computed. This has two main advantages: a loss of significance is less likely, since numbers near zero in the posteriori map to negative values in the log-posteriori, and products are avoided, since the log-posteriori is the sum of several log-probabilities.
- Equations (5.5)-(5.8) are used, where possible, in the application of the algorithm, since they are clearly less expensive than (5.4): these optimizations cut the execution time from several hours to roughly 25 minutes for 10^4 iterations. The optimized C++ code for the algorithm is available in Appendix A.1.3.
- A burn in of 2000 is set and other 20000 values are sampled in order to make the distributions of the parameters as accurate as possible.

In a MCMC algorithm the first thing to check is whether the Markov chain converges to a unique stationary distribution. Convergence will be shown by plotting the evolution of 3 parameters (a mean, a variance and an ε), given two different initial conditions.

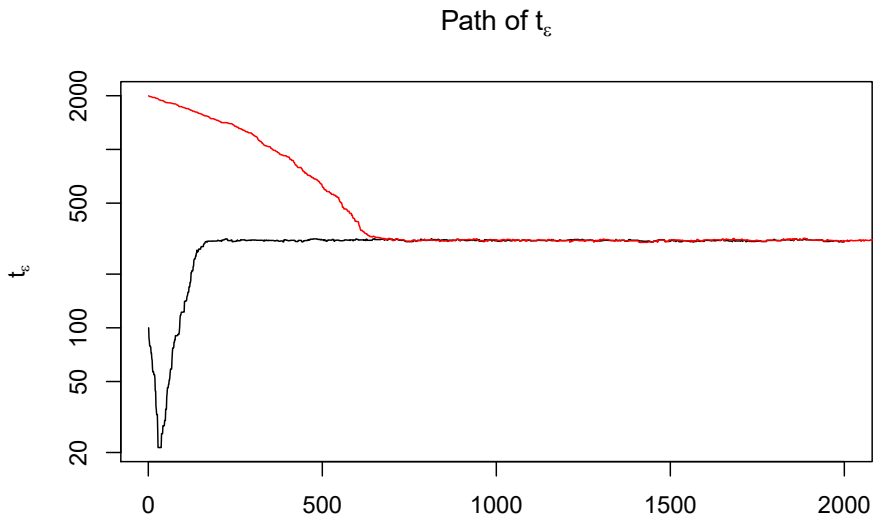

Figure 5.1: Evolution of parameter t_ε

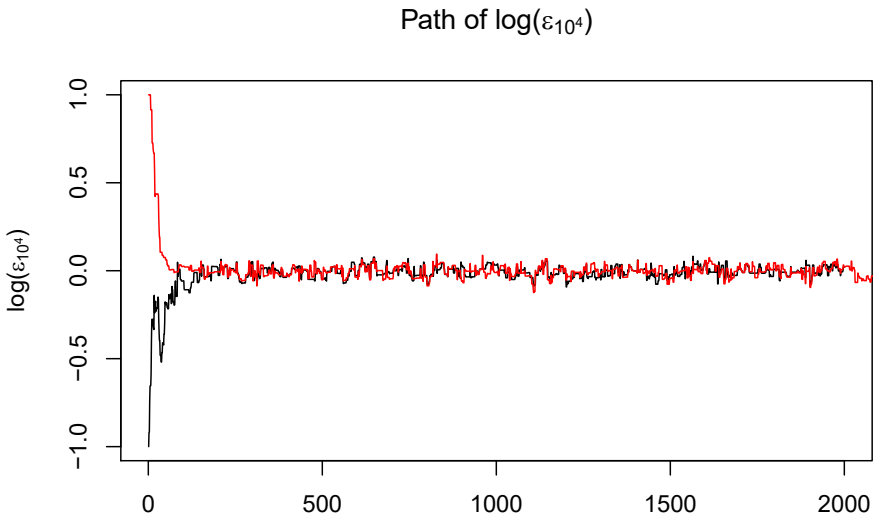

Figure 5.2: Evolution of parameter ε_{10^4}

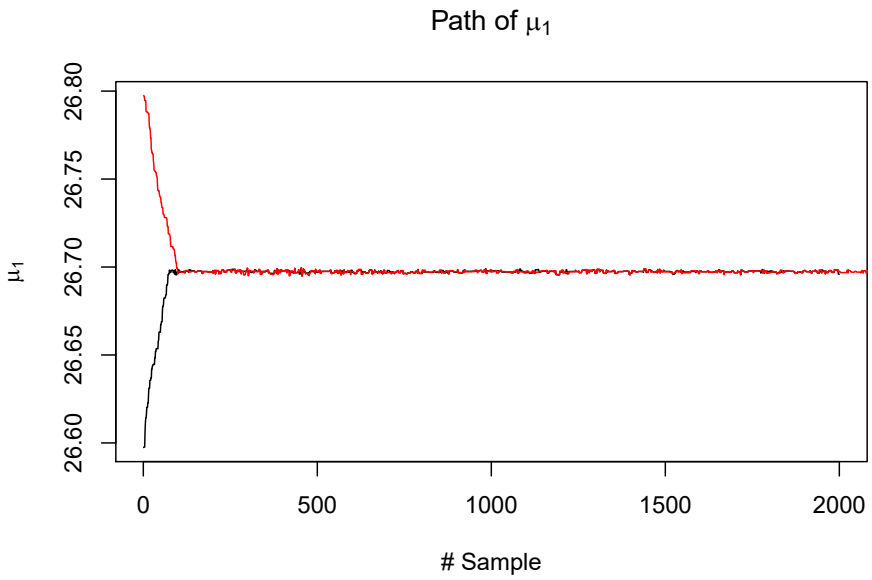

Figure 5.3: Evolution of parameter μ_1

Figures 5.1 5.2 and 5.3 show the evolutions of the Markov chains: the two paths (one drawn in black, the other in red) always converge to the same stationary distribution.

Once convergence is assessed, with 2000 samples that represent a sufficient burn-in period, the acceptance rates (saved in `acc1.csv`) are checked in order to be sure that they stay in an acceptable range. Since the acceptance rates are ≈ 65000 only the minimum and the maximum value are shown.

```
acc<-as.numeric(read.table("acc1.csv",header = T)[,1])/1.2e4
c(min(acc),max(acc))
```

```
## [1] 0.2220833 0.3803333
```

The minimum value is sufficiently near 0.234 and the maximum value is below 40%: these values are totally acceptable.

5.3.1 Parameter Posteriori Distributions

In this bayesian setting it is possible to retrieve the a-posteriori distribution for the parameters, some of them are plotted in order to get a clearer view of the model.

MCMC Distributions of the Means

The aim is now to inspect the a-posteriori distributions of the means.

The first thing to inspect is how variable they are. The 28 distributions are not expected to have the same variance, since they are means of segments of different lengths. The variability of the mean is expected to be decreasing as the segment length grows, thus the distributions of the means of the longest (n.11) and the shortest (n. 17) segment are plotted. The black vertical line represents the approximate maximum likelihood estimates of the parameter.

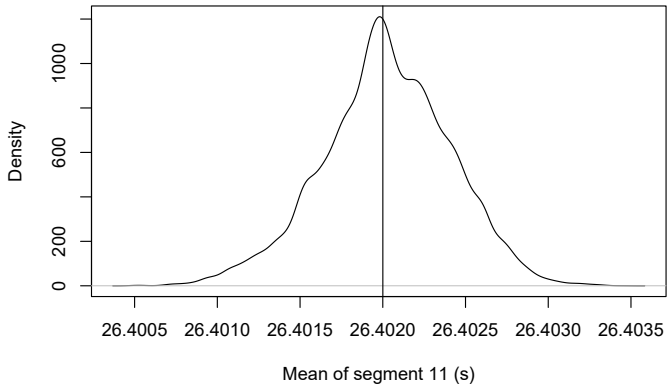

Figure 5.4: Mean of segment 11 MCMC distribution

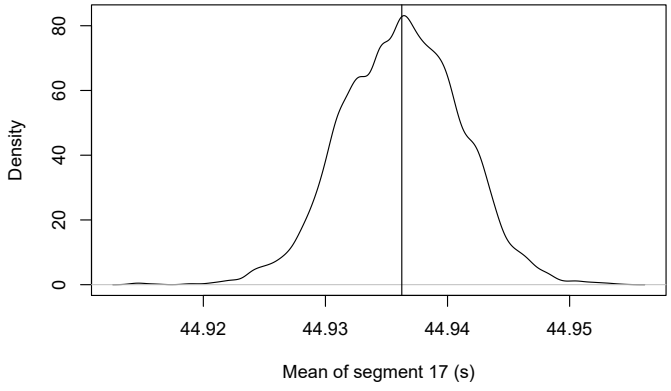

Figure 5.5: Mean of segment 17 MCMC distribution

As expected the MCMC distribution of the mean of the shortest segment has a range of more than 0.03s, while the longest segment's mean's range is roughly 10 times less; these two plots also show that, at least for these two means, the maximum likelihood estimates fall in the range of the a-posteriori distributions.

Since the bayesian model has distributions as outputs, it is possible to look for statistically equal means.

First a plot of the distances between the (sorted) means is shown in order to see if some difference between the ML means are comparable to the ranges of the distributions. The minimum of these difference is shown below.

```
mus=data.frame(seg=1:28,length=diff(c(0,cpts)),mu=model.approx.par$mu)
min(diff(mus[order(mus$mu),]$mu))
```

```
## [1] 0.001209955
```

```
plot(diff(mus[order(mus$mu),]$mu),log="y",xaxt='n', ann=FALSE)
```

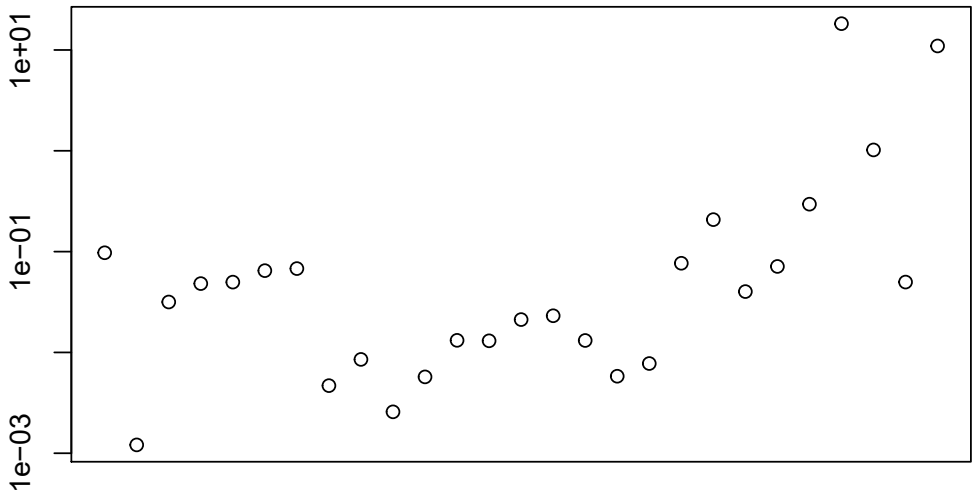

Figure 5.6: Distances between sorted ML means

The means with the smallest distance (roughly 10^{-3}) are μ_6 and μ_9.

In order to see how similar these distributions are, a density plot of the distribution of μ_9 is shown in black, while μ_6 is shown in red. The vertical lines, as seen above, show the maximum approximated likelihood estimate of the means.

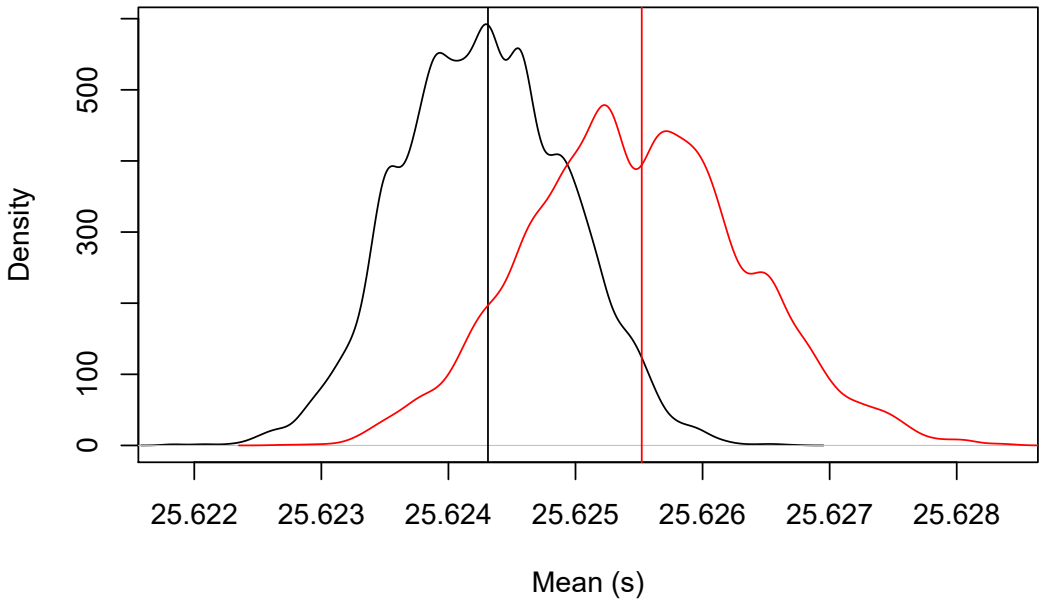

Figure 5.7: Posterior densities of the two means with lowest distance

The two distributions overlap for a good interval, so these two means might be equal according to their a-posteriori distributions.

By observing Figure 5.6 it is possible to see two groups of means distanced by less than 10^{-2}. The first group considered is formed by 5 means, which are plotted in increasing order (according to their ML estimates) in black, along with the immediately greater mean (in red) in order to see if some overlaps are present.

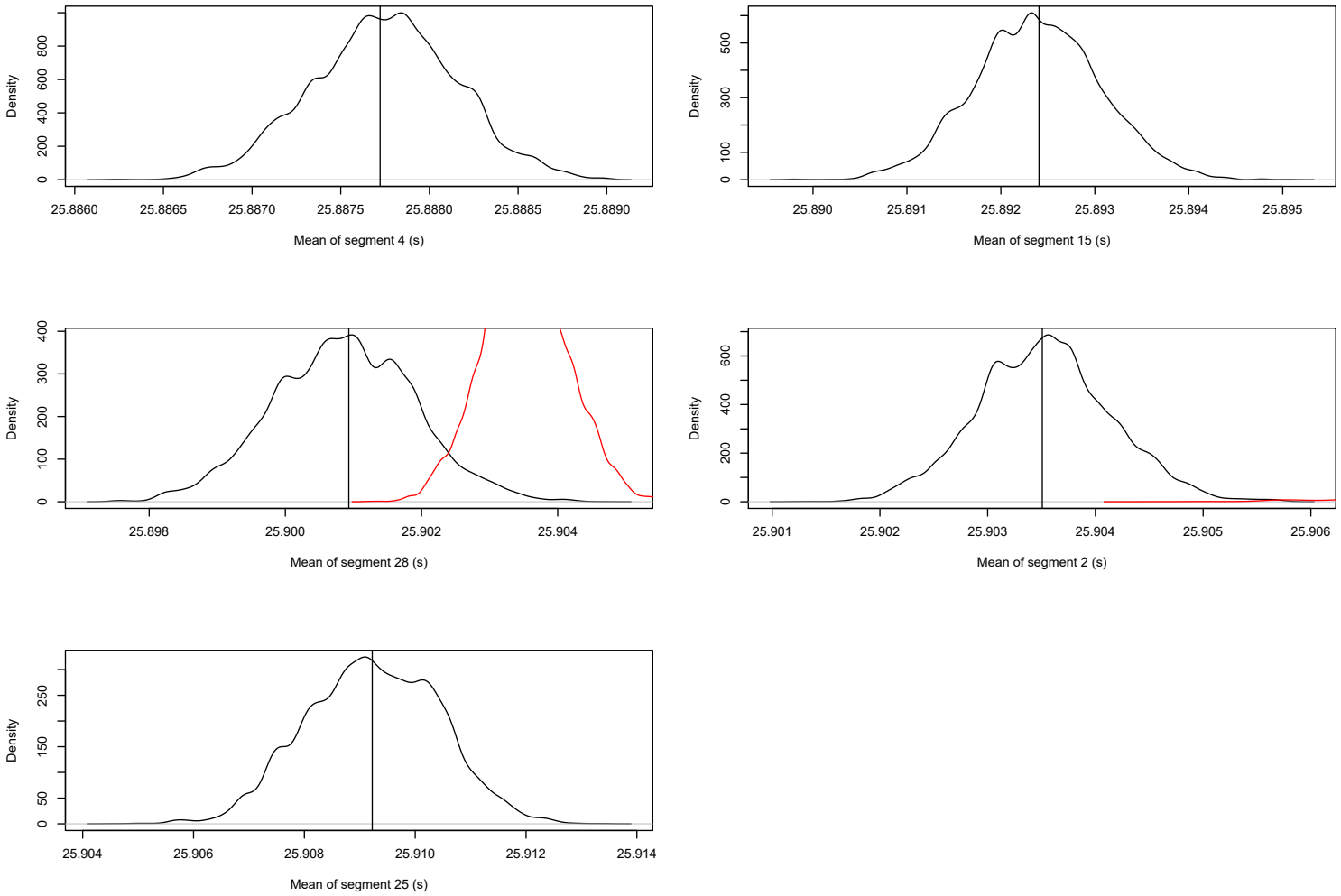

Figure 5.8: Compared densities of low-distanced means (first group)

By looking at the plots the only evidence of a significant overlapping in this group is between the distributions μ_{28} and μ_2.

The second group of low-distanced means is now considered. The plots follow the same logic as in Figure 5.8.

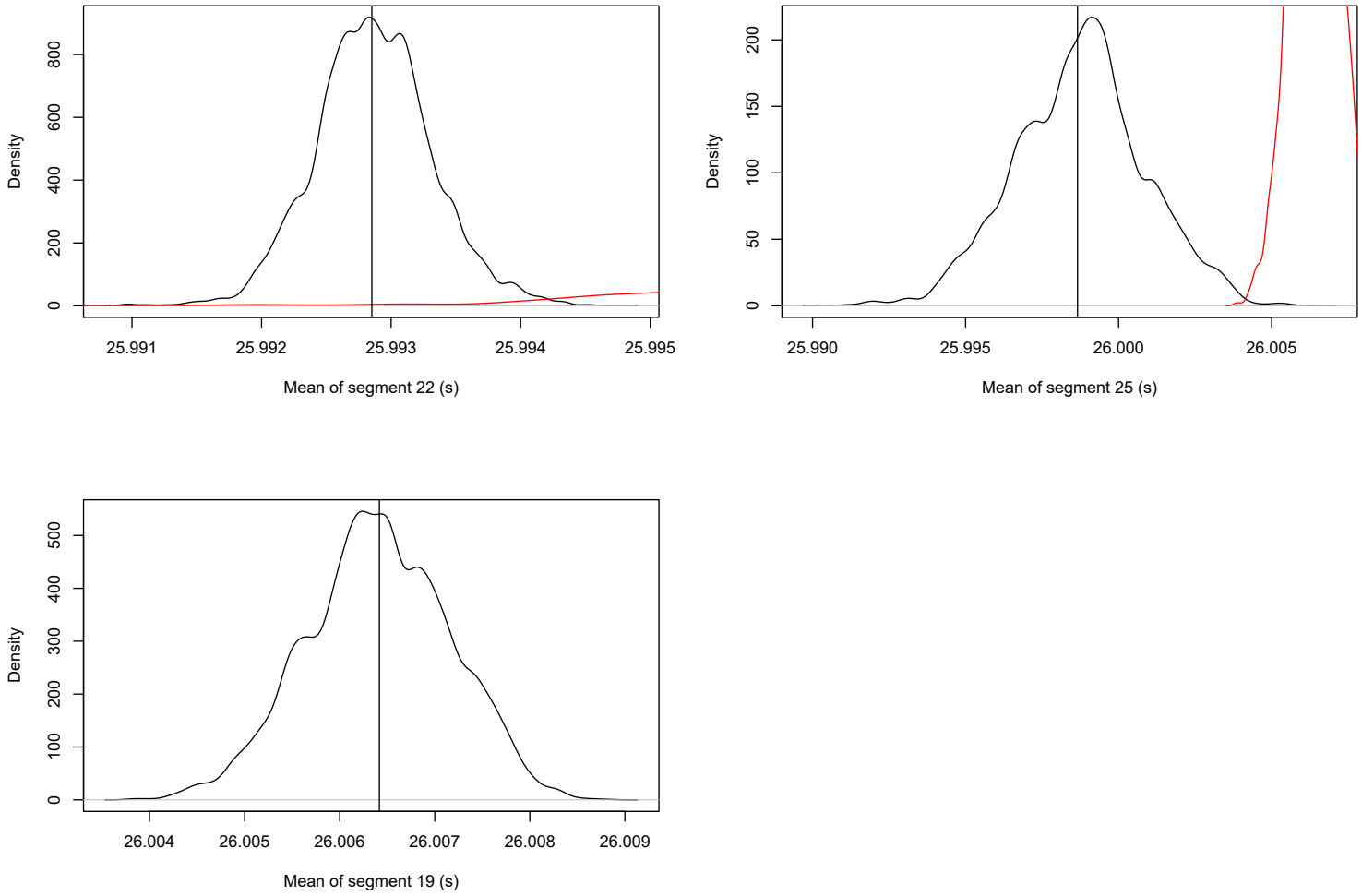

Figure 5.9: Compared densities of low-distanced means (second group)

It can be seen that in this group no significant overlaps are present. As the distances grow it becomes unlikely to observe two overlapping a-posteriori distribution. Even if the distances between the means may appear insignificant, they can't be considered equal with the exceptions of two pairs.

It can be also interesting to notice that the approximated maximum likelihood estimates are at the peak of all the 11 a-posteriori distribution of the means.

MCMC Distributions of Measurement Delays

The considered hierarchical model computes the distributions of every realization of the random variable ε, which represents the impact of measurement delay.

The realization number 10^4 is plotted, as an example, in order to see how precisely the measurement delays are estimated in the model.

```
plot(density(exp(out1$loge10k[(burn+1):K])),main="",xlab="Epsilon (#10000)")
```

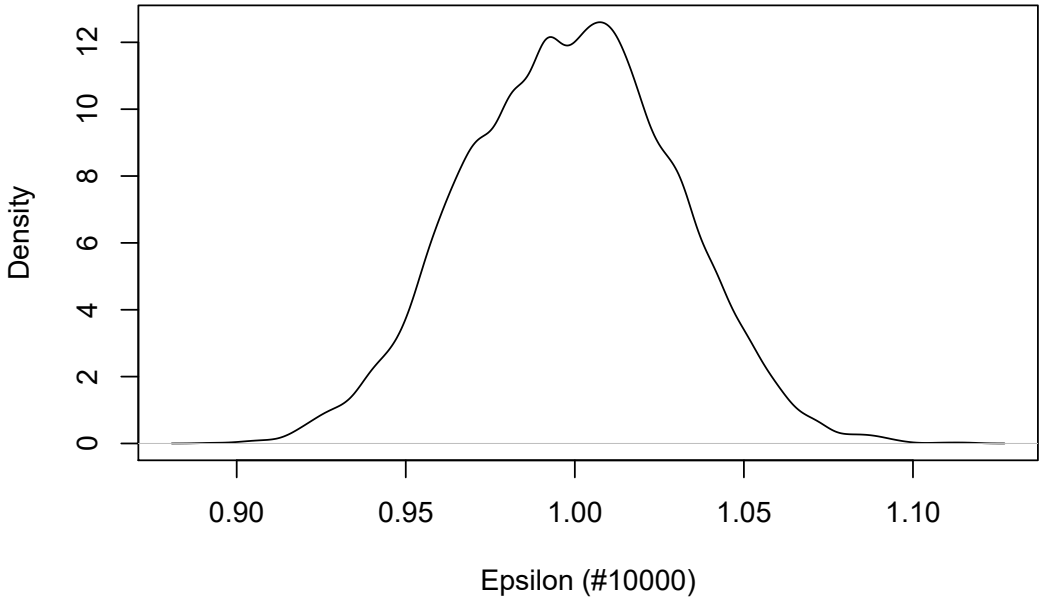

Figure 5.10: MCMC Distribution of the realization #10000 of Epsilon

The distribution of this realization of ε has mean 1, which is the mean value arbitrarily decided when the model has been defined. There are some realizations of ε that differ from the mean, like the one plotted below.

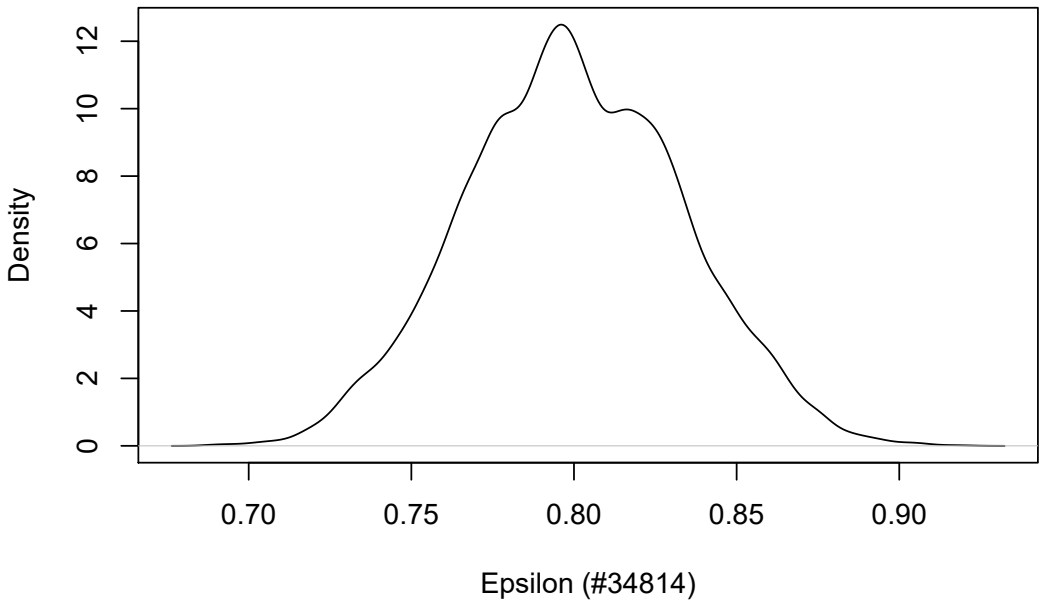

Figure 5.11: MCMC Distribution of the realization #34814 of Epsilon

The realization ε_{38414} represents a below-average measurement delay in the process.

MCMC Distributions of the Variances

The model gives the distributions of two variances: σ^2, related to the actual process $\boldsymbol{X}$, and σ_ε^2, related to the measurement delays $\boldsymbol{\varepsilon}$.

The bayesian distribution of σ^2 is plotted and compared to its approximate maximum likelihood estimate (black vertical line).

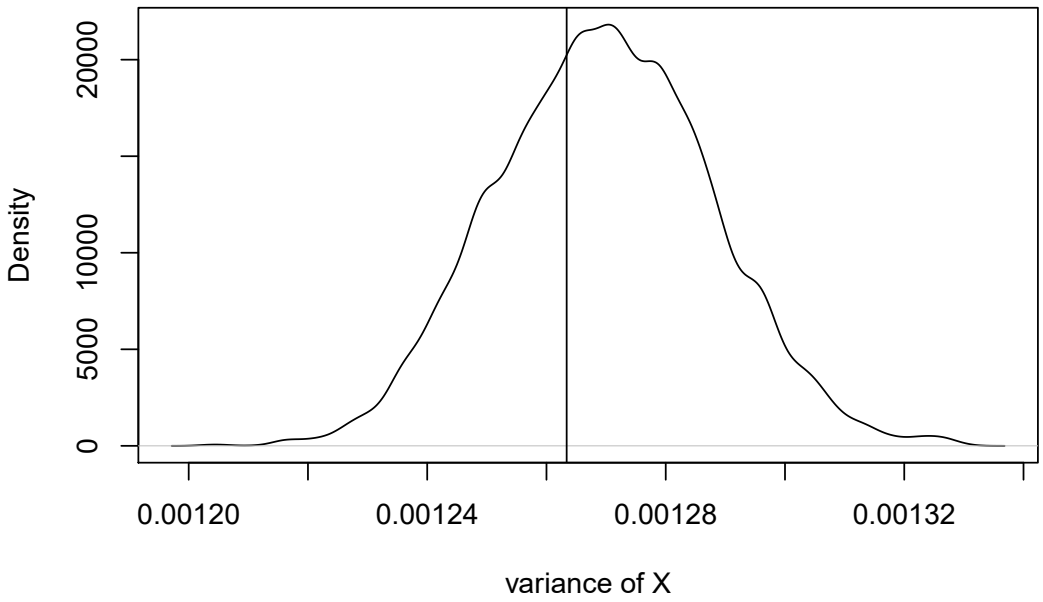

Figure 5.12: Variance of X MCMC distribution

It can be seen that the vertical line is almost at the peak of the plotted distribution, meaning that the bayesian model and the approximate frequentist model give similar estimates of σ^2.

The same cannot be said for σ_ε^2: the maximum likelihood estimate of the parameter under the approximate model is recalled.

```
model.approx.par$sigma2e
```

```
## [1] 0.003066674
```

The a-posteriori distribution of the parameter is now plotted.

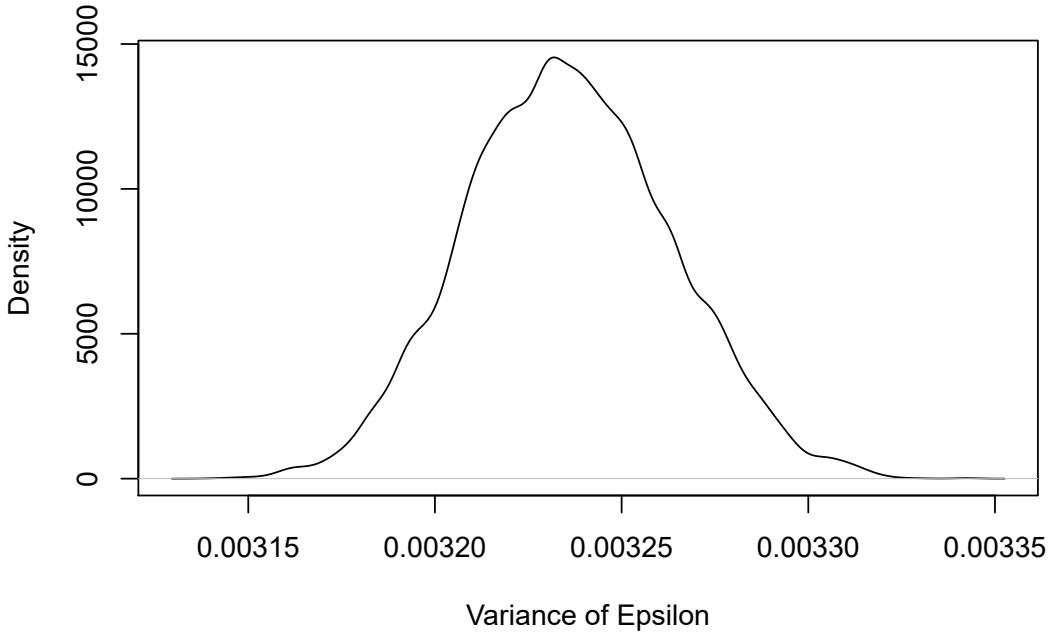

Figure 5.13: Variance of epsilon MCMC distribution

The minimum value of the sampled a-posteriori is greater than the approximate frequentist estimate, hence one of these estimates may not be accurate.

In order to better inspect the accuracy of the estimates, it can be interesting to see how the estimates perform in relation to the total variance.

The variance of Y according to the bayesian model is plotted, the red vertical line is the sample variance of the centered imputated series, while the black vertical line represents the variance calculated as function of the maximum likelihood estimates $\hat{\sigma}^2 + 2\hat{\sigma}_\varepsilon^2$

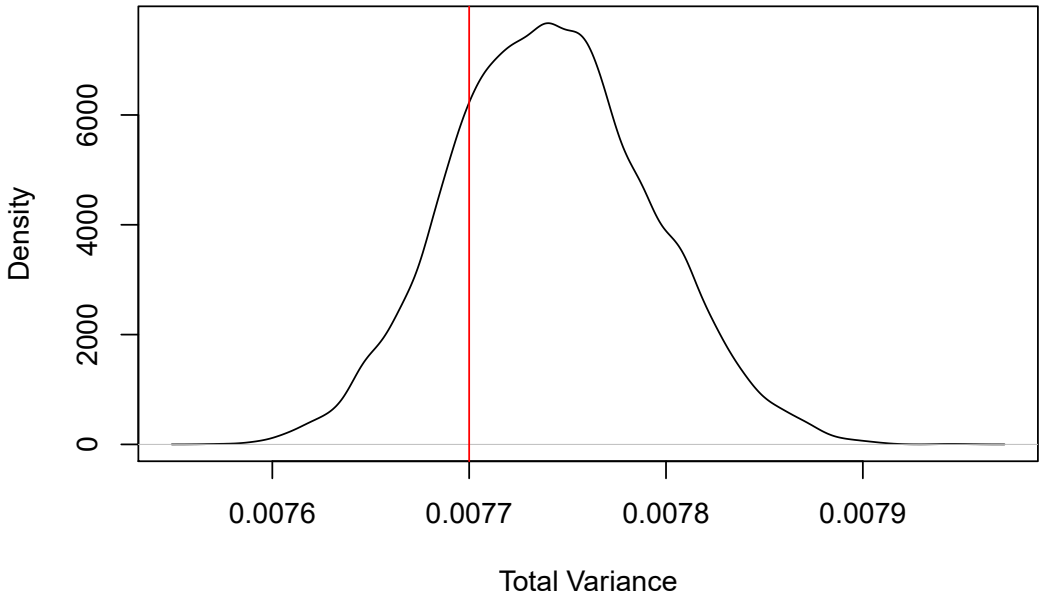

Figure 5.14: Total variance MCMC distribution

The black vertical line falls in the range of the a-posteriori distribution, while no black line is seen.

The variances from the approximated model and the sample variance are shown below

```
data.frame(var_ml=model.approx.par$sigma2+2*model.approx.par$sigma2e,
           var_sample=var(data.c))
```

```
##        var_ml  var_sample
## 1 0.007396734 0.007700001
```

The approximated model estimate of the variance goes below the range of the bayesian distribution, meaning that it is very likely that the model, as described in 3.3, underestimates the parameter σ_ε^2, while the bayesian model gives a more

accurate estimate.

Another measure to be taken into account is the lag-1 autocorrelation, in order to see how well the model captures the impact of the measurement delay.

The distribution of the lag-1 AC is plotted below, the red vertical line represents the sample autocorrelation for the centered,imputated data, while the black vertical line represent the autocorrelation as function of the variances calculated by the approximate maximum likelihood estimation.

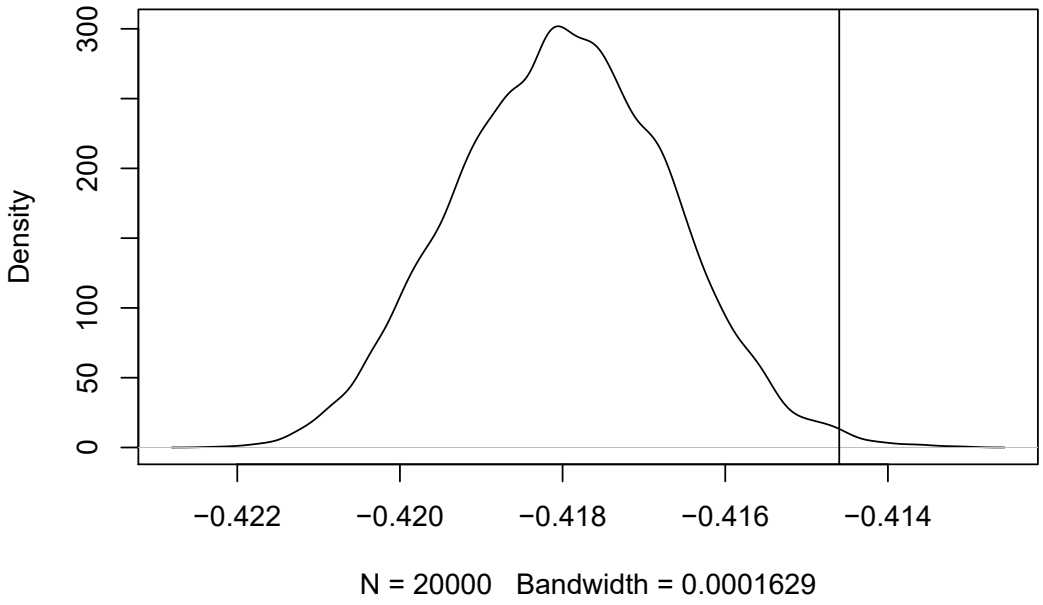

Figure 5.15: Lag-1 Autocorrelation MCMC distribution

The ML estimation of the lag-1 autocorrelation falls in the right tail of the distribution, which is expectable since the parameter σ_ε^2 is underestimated in that model.

The sample lag-1 autocorrelation, which is recalled below, falls out of the range of the distribution.

```
data.frame(ac1_ml=-model.approx.par$sigma2e/
             (2*model.approx.par$sigma2e+model.approx.par$sigma2)
           ,ac1_sample=acf(data.c,plot=F)$acf[2])
```

```
##       ac1_ml ac1_sample
## 1 -0.4145984 -0.4538356
```

The lag-1 autocorrelation is overestimated by both models, nevertheless the bayesian model seems again to be more accurate, as it represent an improvement with respect to the frequentist approach. The bayesian model is also more widely applicable, since it does not rely on normal approximation, which in some cases cannot be done.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The data analysis done in this thesis for the cycle times shows that a mean value change-point model robust to extreme outliers is needed in order to better detect anomalies in the manufacturing process. The cycle time measured may present values that are several times higher or lower than expected and it is strongly recommended to check, if actually produced, the corresponded products; some anomalies "balance" each other, meaning that an anomaly in the measurement system cannot be excluded as a cause.

Once the outliers are detected and possibly removed, a negative autocorrelation at lag 1 is detected, since the actual variability of cycle times resulted to be lower than the variability induced by the measurement system.

Two models have been developed in order to capture the impact of the variability of the measurement system: the approximated frequentist one is to prefer when the aim is to have a fast implementation at expenses of some precision, while the Bayesian one, despite it is slower computationally, makes no approximation and gives more encouraging results.

The lag-1 autocorrelation is not captured perfectly by both models, a possible reason for this can be that in each segment the mean value may not be perfectly constant due to various reasons. The extremely low variability in the process and the fact that the model assumes a constant value of μ for each segment may cause the parameter σ^2 to be inflated.

6.2 Future Work

The model gives good qualitative results in capturing the change-points and the variance induced by the measurement error, though a possible (and more computationally expensive) improvement could be to consider, instead of segments with constant means, some very small trends or fluctuations within each interval, this can be useful in order to better capture the autocorrelation of the cycle times.

Although the dataset is not small, some other improvements to the model that require more data are presented. Since most of the mean values are in a limited range (25 to 27 seconds), a way to improve it could be a clustering of the segments. The starting point would be the model in (4.2), with the hypothesis that μ_j is a random variable with a cumulative distribution $F_{M_h}(x)$ dependent on a cluster M_h.

Two possible hierarchical clusterings of the segments are shown, the former using the Dynamic Time Warp distance for time series [31], implemented in the package `TSdist` [32], and the latter using the euclidean distance for the means found using the frequentist model.

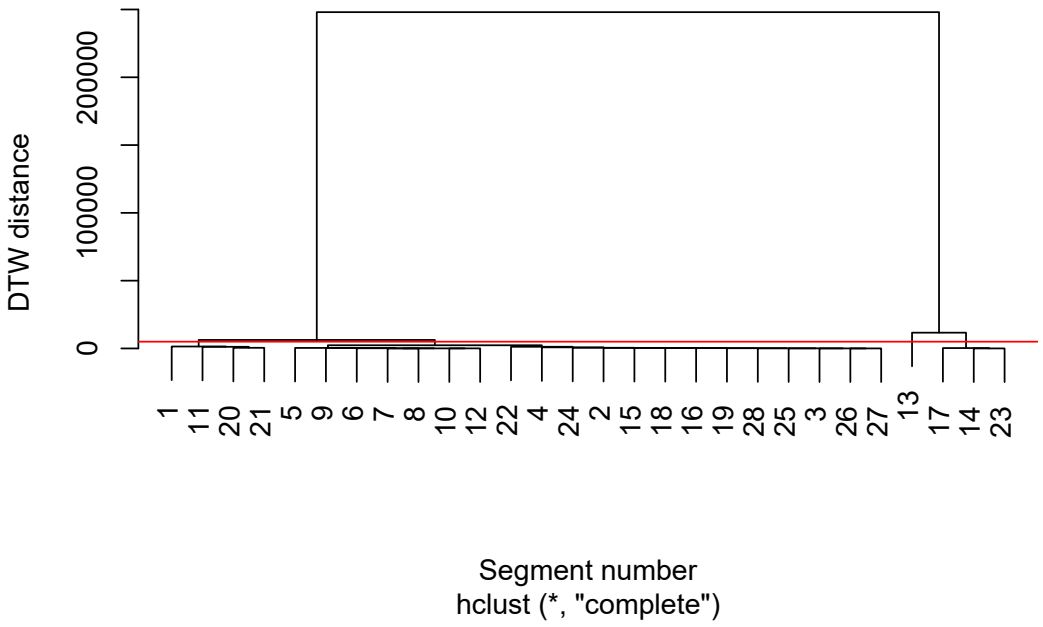

Figure 6.1: Hierarchical Clustering of the segments using Dynamic Time Warp distance

Figure 6.1 shows that most segments have small distances between them. A red horizontal line that splits the segments into four clusters is drawn. If the model is accurate then a clustering on the means will give similar results: a dendrogram using the euclidean distance between the means is plotted in figure 6.2.

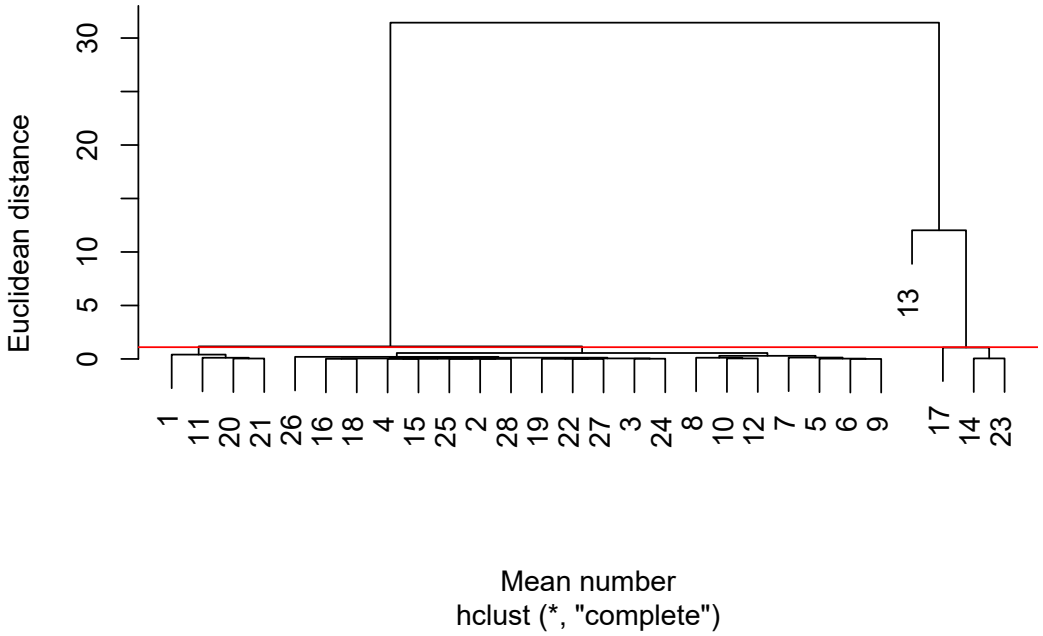

Figure 6.2: Hierarchical Clustering of the segments using Euclidean distance between ML means

The plot shows that, choosing a suitable threshold (represented again by a red horizontal line), it is possible to obtain the same clusters as in the previous setting, meaning that a clustering made by only considering distances in the segment means should not lose much accuracy.

However a more detailed analysis requires more segments (hence more data) since a clustering on 28 points is heavily inaccurate due to small sample size.

Once the clustering is done, another improvement may be the construction of a more complex model: the clusters can be interpreted as states of a hidden Markov chain Γ_M where a transition is made at each change-point.

The behaviour of the cluster would be represented by the equation

$$P[\mu_{j+1} \in M_k | \mu_j \in M_h] = \Gamma_M(h,k)$$

having that

$$\mu_j \in M_h \iff P[\mu_j \leq x] = F_{M_h}(x)$$

A further improvement can be to suppose, for each cluster, a segment length distribution.

$$P[\tau_{j+1} - \tau_j | \mu_j \in M_h] = p^{(\tau)}_{M_h}(x)$$

The implementation of the model requires a huge amount of data from the manufacturing process, especially in order to get some accurate estimates of the transition probabilities in Γ_M.

The improvements shown above require, besides more data, a huge computational time, since another level would be added to the hierarchical model.

Appendix A

Functions

A.1 List of C++ functions

The functions coded below are written in C++ language and implemented in R using the package `Rcpp`. Information and manuals on Rcpp are available on [33][34][35].

A.1.1 Log-Likelihood Algorithm

The following function computes the approximate log-likelihood as described in section 3.3. The maximization of the log-likelihood can be made with functions such as `nlm`.

```
#include <Rcpp.h>
using namespace Rcpp;
double ll_iet_l(List l,double s2,double s2e){
            double res=0;

            for(int h=0;h<l.size();h++){
            DataFrame df=l[h];
            NumericVector delta=df["iet"];
            NumericVector mu=df["mu"];

            int n=delta.size();
            int i;
            if(mu.size()!=n)
            stop("Error: data and means have different lengths");
            NumericVector xi=delta-mu;
            double al0n=0.5*(1/s2+1/s2e);
```

```
17              double ali=1/s2+0.5/s2e;
18              double gamma=-1/s2;
19
20              NumericVector beta (n+1);
21              NumericVector a(n+1);
22              NumericVector b(n+1);
23              NumericVector c(n+2);
24              a[0]=al0n;
25              beta[0]=xi[0]/s2;
26              b[0]=beta[0];
27              c[0]=sum(pow(xi,2))/(2*s2);
28              for(i=1;i<n;i++){
29              a[i]=ali-pow(gamma,2)/(4*a[i-1]);
30              beta[i]=(xi[i]-xi[i-1])/s2;
31              }
32              a[n]=al0n-pow(gamma,2)/(4*a[n-1]);
33
34              beta[n]=-(xi[n-1])/s2;
35
36
37              for(i=1;i<=n;i++)
38              {
39
40              b[i]=beta[i]-gamma*b[i-1]/(2*a[i-1]);
41              c[i]=-pow(b[i-1],2)/(4*a[i-1]);
42              }
43
44              c[n+1]=-pow(b[n],2)/(4*a[n]);
45
46
47              res+=-0.5*n*log(s2)-0.5*(n+1)*log(s2e)-0.5*sum(log(a))-
    sum(c);
48
49              }
50              return(res);
51              }
```

A.1.2 Adjusted CUSUM algorithm

The following function computes the adjusted CUSUM statistic as described in [26].

```
#include <Rcpp.h>
using namespace Rcpp;
    NumericVector cusum_adj(NumericVector x){
            int N=x.size();
            NumericVector cus(N);
            NumericVector k(N);
            double theta=mean(x);
            double sigma2=var(x);
            int i;
            cus[0]=x[0]-theta;
            for(i=1; i<N; i++){
            cus[i]=cus[i-1]+(x[i]-theta);
            k[i-1]=i*(N-i)/(pow(N,2));
            }
            k[N-1]=1;

            NumericVector res = pow(cus,2)/(k*sigma2*N);
            return(res);

}
```

A.1.3 MCMC algorithm for Bayesian Model

The code shown below is an implementation of the Metropolis within Gibbs algorithm to the hierarchical model presented in Chapter 5. The function `posteriori` is called only when means and variances are evaluated in order to speed up the execution of the code.

```
#include <Rcpp.h>
using namespace Rcpp;

double posteriori(NumericVector d, NumericVector x,IntegerVector ooc,
    IntegerVector ooc_e,NumericVector cp, int ind){
int ncp=cp.size();
int eind;
int j=0;
int i;
int ntot=x.size();
int ne=ntot-ncp-2;
int ndata=d.size();
```

```
12 int start;
13 int end;
14
15 if(ndata+max(ooc)!=ne) stop("Error:Cannot formulate an aPosteriori.
       Vectors not compatible");
16
17 NumericVector mx(ncp);
18 NumericVector mln(ncp);
19 NumericVector sdln(ncp);
20 double tx=x[ntot-2];
21 double te=x[ntot-1];
22 double x_i;
23 double mlne=-0.5/sqrt(te);
24 double sdlne=1/sqrt(te);
25 double res=0;
26
27 if (ind<ne+ncp){
28 if(x[ind]<=0) return R_NegInf;
29 res=res+R::dnorm(log(x[ind]),3,2,true);
30 if(ind==ne) start=0;
31 else start=cp[ind-ne-1];
32 end=cp[ind-ne];
33 for(i=start;i<end;i++){
34   eind=ooc[i];
35     x_i=d[i]-exp(x[i+eind])+exp(x[i+eind-1]);
36     if(x_i<=0) return R_NegInf;
37     res=res+R::dnorm(log(x_i),2*log(x[ind])-0.5*log(pow(x[ind],2)+1/
       tx),sqrt(log(1+1/(pow(x[ind],2)*tx))),true);
38 }
39 }
40 else
41 {
42 for(i=0; i<ncp;i++){
43 mx[i]=x[ne+i];
44 if(mx[i]<=0) return R_NegInf;
45 res=res+R::dnorm(log(mx[i]),3,2,true);
46 mln[i]=2*log(mx[i])-0.5*log(pow(mx[i],2)+1/tx);
47 sdln[i]=sqrt(log(1+1/(pow(mx[i],2)*tx)));
48 }
49 for(i=0;i<ne;i++){
50 if(i<ndata){
51 if(i==cp[j]) j=j+1;
52     eind=ooc[i];
53     x_i=d[i]-exp(x[i+eind])+exp(x[i+eind-1]);
54     if(x_i<=0) return R_NegInf;
55     res=res+R::dnorm(log(x_i),mln[j],sdln[j],true);
56 }
57 res=res+R::dnorm(x[i],mlne,sdlne,true);
58
```

```
59 }
60
61
62 res=res+R::dgamma(tx,1e-4,1e4,true)+R::dgamma(te,1e-4,1e4,true);
63 }
64
65 return res;
66 }
67
68 //' @export
69 //[[Rcpp::export]]
70 List mwg_cp(int K,int burnin,NumericVector d,IntegerVector ooc,
       NumericVector cp,NumericVector inits,NumericVector propsd){
71     int ntot=inits.size();
72     NumericMatrix chain(K,ntot);
73     IntegerVector accepted(ntot);
74     double cand_post;
75     double old_post;
76     double log_acc_ratio;
77     double oldpar;
78     double x_i;
79     double x_old;
80     NumericVector old_vec(ntot);
81     int eind;
82     int seg;
83     double mlne;
84 double sdlne;
85 int ndata=ooc.size();
86 if(ndata!=d.size()) stop("Error: data and oocs lengths differ");
87 int n_oo=ooc[ndata-1];
88 int ne=ndata+n_oo;
89
90 int oo=1;
91 IntegerVector ooc_e(ne);
92 for(int i=0;i<ndata;i++){
93   if(ooc[i]!=oo)
94   {ooc_e[i+oo-1]=oo;}
95   oo=ooc[i];
96   ooc_e[i+oo-1]=oo;
97 }
98 ooc_e[ne-1]=n_oo;
99
100
101     for(int k=0;k<ntot;k++){chain(0,k)=inits[k];}
102
103     for(int j=1;j<K;j++){
104     chain(j,_)=chain(j-1,_);
105     seg=0;
106         for(int i=0;i<ntot;i++){
```

```
107         oldpar=chain(j,i);
108             chain(j,i)=oldpar+R::rnorm(0,propsd[i]);
109
110             if(i<ne){
111             mlne=-0.5/sqrt(chain(j,ntot-1));
112             sdlne=1/sqrt(chain(j,ntot-1));
113             eind=ooc_e[i];
114             if (i-eind==cp[seg]) seg=seg+1;
115
116             cand_post=0;
117             old_post=0;
118             if(i==0 || ooc_e[i-1]!=ooc_e[i]){
119             x_i=d[i-eind+1]-exp(chain(j,i+1))+exp(chain(j,i));
120             x_old=d[i-eind+1]-exp(chain(j,i+1))+exp(oldpar);
121             if(x_i<=0 || x_old<=0)
122             {if (x_i<=0) cand_post=R_NegInf;
123             if (x_old<=0) old_post=R_NegInf;
124             }
125             else{
126             if(i-eind+1==cp[seg])
127             {cand_post=cand_post+R::dnorm(log(x_i),2*log(chain(j,ne+
    seg+1))-0.5*log(pow(chain(j,ne+seg+1),2)+1/chain(j,ntot-2)),sqrt(
    log(1+1/(pow(chain(j,ne+seg+1),2)*chain(j,ntot-2)))),true);
128             old_post=old_post+R::dnorm(log(x_old),2*log(chain(j,ne+
    seg+1))-0.5*log(pow(chain(j,ne+seg+1),2)+1/chain(j,ntot-2)),sqrt(
    log(1+1/(pow(chain(j,ne+seg+1),2)*chain(j,ntot-2)))),true);
129             }
130             else{
131             cand_post=cand_post+R::dnorm(log(x_i),2*log(chain(j,ne+
    seg))-0.5*log(pow(chain(j,ne+seg),2)+1/chain(j,ntot-2)),sqrt(log
    (1+1/(pow(chain(j,ne+seg),2)*chain(j,ntot-2)))),true);
132             old_post=old_post+R::dnorm(log(x_old),2*log(chain(j,ne+
    seg))-0.5*log(pow(chain(j,ne+seg),2)+1/chain(j,ntot-2)),sqrt(log
    (1+1/(pow(chain(j,ne+seg),2)*chain(j,ntot-2)))),true);
133             }
134             }
135             }
136             else if (i==ne-1 || ooc_e[i+1]!=ooc_e[i]){
137             x_i=d[i-eind]-exp(chain(j,i))+exp(chain(j,i-1));
138             x_old=d[i-eind]-exp(oldpar)+exp(chain(j,i-1));
139             if(x_i<=0 || x_old<=0)
140             {if (x_i<=0) cand_post=R_NegInf;
141             if (x_old<=0) old_post=R_NegInf;
142             }
143             else {cand_post=cand_post+R::dnorm(log(x_i),2*log(chain(j
    ,ne+seg))-0.5*log(pow(chain(j,ne+seg),2)+1/chain(j,ntot-2)),sqrt(
    log(1+1/(pow(chain(j,ne+seg),2)*chain(j,ntot-2)))),true);
```

```
144             old_post=old_post+R::dnorm(log(x_old),2*log(chain(j,ne+
    seg))-0.5*log(pow(chain(j,ne+seg),2)+1/chain(j,ntot-2)),sqrt(log
    (1+1/(pow(chain(j,ne+seg),2)*chain(j,ntot-2)))),true);
145             }
146             }
147             else{
148             x_i=d[i-eind]-exp(chain(j,i))+exp(chain(j,i-1));
149             x_old=d[i-eind]-exp(oldpar)+exp(chain(j,i-1));
150             if(x_i<=0 || x_old<=0)
151             {if (x_i<=0) cand_post=R_NegInf;
152             if (x_old<=0) old_post=R_NegInf;
153             }
154             else{
155             cand_post=cand_post+R::dnorm(log(x_i),2*log(chain(j,ne+
    seg))-0.5*log(pow(chain(j,ne+seg),2)+1/chain(j,ntot-2)),sqrt(log
    (1+1/(pow(chain(j,ne+seg),2)*chain(j,ntot-2)))),true);
156             old_post=old_post+R::dnorm(log(x_old),2*log(chain(j,ne+
    seg))-0.5*log(pow(chain(j,ne+seg),2)+1/chain(j,ntot-2)),sqrt(log
    (1+1/(pow(chain(j,ne+seg),2)*chain(j,ntot-2)))),true);
157
158             x_i=d[i-eind+1]-exp(chain(j,i+1))+exp(chain(j,i));
159             x_old=d[i-eind+1]-exp(chain(j,i+1))+exp(oldpar);
160             if(x_i<=0 || x_old<=0)
161             {if (x_i<=0) cand_post=R_NegInf;
162             if (x_old<=0) old_post=R_NegInf;
163             }
164             else{
165             if(i-eind+1==cp[seg])
166             {cand_post=cand_post+R::dnorm(log(x_i),2*log(chain(j,ne+
    seg+1))-0.5*log(pow(chain(j,ne+seg+1),2)+1/chain(j,ntot-2)),sqrt(
    log(1+1/(pow(chain(j,ne+seg+1),2)*chain(j,ntot-2)))),true);
167             old_post=old_post+R::dnorm(log(x_old),2*log(chain(j,ne+
    seg+1))-0.5*log(pow(chain(j,ne+seg+1),2)+1/chain(j,ntot-2)),sqrt(
    log(1+1/(pow(chain(j,ne+seg+1),2)*chain(j,ntot-2)))),true);
168
169             }
170             else{
171           cand_post=cand_post+R::dnorm(log(x_i),2*log(chain(j,ne+seg)
    )-0.5*log(pow(chain(j,ne+seg),2)+1/chain(j,ntot-2)),sqrt(log(1+1/(
    pow(chain(j,ne+seg),2)*chain(j,ntot-2)))),true);
172             old_post=old_post+R::dnorm(log(x_old),2*log(chain(j,ne+
    seg))-0.5*log(pow(chain(j,ne+seg),2)+1/chain(j,ntot-2)),sqrt(log
    (1+1/(pow(chain(j,ne+seg),2)*chain(j,ntot-2)))),true);
173
174             }
175
176             }
177             }
178             }
```

```
179                 cand_post=cand_post+R::dnorm(chain(j,i),mlne,sdlne,true);
180                 old_post=old_post+R::dnorm(oldpar,mlne,sdlne,true);
181                 }
182
183                 else {cand_post=posteriori(d,chain(j,_),ooc,ooc_e,cp,i);
184                 old_vec=chain(j,_);
185                 old_vec[i]=oldpar;
186                 old_post=posteriori(d,old_vec,ooc,ooc_e,cp,i);
187                 }
188
189
190                 log_acc_ratio=cand_post-old_post;
191
192                 if(log_acc_ratio>log(R::runif(0,1))) {
193                 if(j>burnin) accepted[i]++;
194                 }
195                 else {
196                 chain(j,i)=oldpar;
197                 }
198             }
199         }
200         return List::create(chain,accepted);
201 }
```

A.2 List of R functions

A.2.1 Robust CUSUM algorithm

This function includes several robust CUSUM statistics. It is called by the function `binseg.rob.cusum`.

```
1     rob_Cusum <- function(X,k=1.5,s0=mad(X),adj=FALSE,est="trimmed",
      tr=0.1,mw=10,r.tol=1e-3){
2   if(est=="trimmed")
3   {K=k*s0
4     theta=mean(X,trim=tr)}
5   else
6     if(est=="median")
7     {K=k*s0
8       theta=median(X)}
9     else
10      if(est=="huberm")
11        {K=k*s0
12        theta=hubermean(X,K)}
13      else
14        if (est=="auto")
```

```
15          {
16            K=k*s0
17            mw=medMW(X,n=mw)
18            Y=sapply(X,function(x) min(max(x,min(mw)-K),max(mw)+K))
19            return(cumsum(Y-mean(Y)))
20          }
21        else
22          if (est=="M"){
23            s.old=m.old=-Inf
24            s.new=mad(X)
25            m.new=median(X)
26            while(abs(s.old-s.new)>r.tol*s.new | abs(m.old-m.new)>r.tol
    *m.new){
27              s.old=s.new
28              m.old=m.new
29            Y=sapply(X,function(x) min(max(x,min(m.old)-k*s.old),max(m.
    old)+k*s.old))
30            s.new=sqrt(var(Y))
31            m.new=mean(Y)
32            }
33            K=k*s.new
34            theta=m.new
35            s0=s.new
36          }
37        else stop("Estimator Unknown")
38
39
40    if(adj) return(list(CUSUM=cusum_adj(sapply(X,function(x) min(max(x,
     theta-K),theta+K))),mu=theta,sigma=s0))
41        else return(list(CUSUM=cumsum(sapply(X,function(x) min(max(x-
    theta,-K),K))),mu=theta,sigma=s0))
42  }
```

A.2.2 Negative Likelihood Function

The following function is fed to the non-linear minimizer `nlm`. Since `nlm` does not put bounds to parameters, the variances have to be transformed with an invertible function f so that

$$f(\sigma^2) : (0, +\infty) \rightarrow \mathbb{R}$$

The vector `parvec` contains the transformed parameters. Every time this code is applied in the thesis the transformation of the variances will be done using $f(x) = log(x)$.

```
1     mlltot<-function(x,cps,parvec){
2
3   par<-model.cp.np(parvec)
4   means=par$mu
5   s2=par$sigma2
6   s2e=par$sigma2e
7
8   cps.diff=diff(c(0,cps))
9   ms<-rep(means,cps.diff)
10
11  isout=which(x$outl)
12
13  if(length(isout)>0)
14  {
15    x$i<-x$i+cumsum(x$outl)
16    x<-x[-isout,]
17    ms<-ms[-isout]
18  }
19  df.in<-data.frame(iet=x$iet,i=x$i,mu=ms)
20  list.in<-split(df.in,df.in$i)
21
22  -ll_iet_l(list.in,s2,s2e)
23
24  #}))
25 }
```

Bibliography

[1] Alexander Dürre, Roland Fried, and Tobias Liboschik. «Robust estimation of (partial) autocorrelation». In: *WIREs Computational Statistics* 7.3 (2015), pp. 205–222. DOI: 10.1002/wics.1351 (cit. on p. 7).

[2] Alexander Dürre, Roland Fried, Tobias Liboschik, and Jonathan Rathjens. *robts: Robust Time Series Analysis*. R package version 0.3.0/r224. 2018 (cit. on p. 7).

[3] Guillem Rigaill. *robseg: Segmentation using functionnal pruning for robust losses*. 2020 (cit. on pp. 12, 44).

[4] F.R. Hampel. *Contributions to the Theory of Robust Estimation*. University of California, 1968 (cit. on p. 23).

[5] Ricardo A Maronna, R. Douglas Martin, Victor J Yohai, and Matías Salibián-Barrera. *Robust Statistics: Theory and Methods (with R)*. eng. Wiley series in probability and statistics. Newark: John Wiley and Sons, Incorporated, 2019. ISBN: 1119214688 (cit. on p. 24).

[6] Peter J. Huber. «Robust Estimation of a Location Parameter». In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101. DOI: 10.1214/aoms/1177703732 (cit. on pp. 24, 25).

[7] Paul Fearnhead and Guillem Rigaill. «Changepoint Detection in the Presence of Outliers». In: *Journal of the American Statistical Association* 114.525 (2019), pp. 169–183. DOI: 10.1080/01621459.2017.1385466 (cit. on pp. 25, 44–46).

[8] Valentin Todorov and Peter Filzmoser. «An Object-Oriented Framework for Robust Multivariate Analysis». In: *Journal of Statistical Software* 32.3 (2009), pp. 1–47. URL: http://www.jstatsoft.org/v32/i03/ (cit. on p. 29).

[9] Martin Maechler et al. *robustbase: Basic Robust Statistics*. R package version 0.93-6. 2020 (cit. on p. 29).

[10] Peter J. Rousseeuw and Christophe Croux. «Alternatives to the Median Absolute Deviation». In: *Journal of the American Statistical Association* 88.424 (1993), pp. 1273–1283. DOI: 10.1080/01621459.1993.10476408 (cit. on p. 29).

[11] Victor J Yohai and Ruben H Zamar. «"High Breakdown-Point Estimates of Regression by Means of the Minimization of an Efficient Scale": TM». In: *Journal of the American Statistical Association* 83.401 (1988), p. 406 (cit. on p. 29).

[12] R. Maronna and R. Zamar. «Robust Estimates of Location and Dispersion for High-Dimensional Datasets». In: *Technometrics* 44 (2002), pp. 307–317 (cit. on p. 30).

[13] C. Spearman. «The proof and measurement of association between two things.» In: *International journal of epidemiology* 39 5 (2010), pp. 1137–50 (cit. on p. 30).

[14] P. Laurie Davies and Ursula Gather. «Breakdown and Groups». In: *The Annals of Statistics* 33.3 (2005), pp. 977–1035 (cit. on p. 31).

[15] R. Gnanadesikan and J. R. Kettenring. «Robust Estimates, Residuals, and Outlier Detection with Multiresponse Data». In: *Biometrics* 28.1 (1972), pp. 81–124 (cit. on p. 31).

[16] Yanyuan Ma and Marc G. Genton. «Highly Robust Estimation of the Autocovariance Function». In: *Journal of Time Series Analysis* 21.6 (2000), pp. 663–684. DOI: https://doi.org/10.1111/1467-9892.00203 (cit. on p. 31).

[17] Robert Maidstone, Toby Hocking, Guillem Rigaill, and Paul Fearnhead. «On Optimal Multiple Changepoint Algorithms for Large Data». In: *Statistics and Computing* 27 (Mar. 2017). DOI: 10.1007/s11222-016-9636-3 (cit. on p. 44).

[18] Kaylea Haynes, Idris Eckley, and Paul Fearnhead. «Efficient penalty search for multiple changepoint problems». In: (Dec. 2014) (cit. on p. 47).

[19] L. Yu. Vostrikova. «Detecting "disorder" in multidimensional random processes». In: *Sov. Math., Dokl.* 24 (1981), pp. 55–59 (cit. on p. 51).

[20] D.C. Montgomery. *Introduction to Statistical Quality Control*. Wiley, 2009. URL: https://books.google.it/books?id=oG1xPgAACAAJ (cit. on p. 52).

[21] M.D. Donsker. «An invariant principle for certain probability limit theorems». In: *Memoirs of the American Mathematical Society* 6 (1951) (cit. on p. 53).

[22] G.R. Shorack and J.A. Wellner. *Empirical Processes with Applications to Statistics*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2009 (cit. on p. 54).

[23] Rebecca Killick and Idris A. Eckley. «changepoint: An R Package for Changepoint Analysis». In: *Journal of Statistical Software* 58.3 (2014), pp. 1–19 (cit. on p. 55).

[24] Rebecca Killick, Kaylea Haynes, and Idris A. Eckley. *changepoint: An R package for changepoint analysis*. 2016. URL: https://CRAN.R-project.org/package=changepoint (cit. on p. 55).

[25] Curtis Miller. *CPAT: Change Point Analysis Tests*. 2018. URL: https://CRAN.R-project.org/package=CPAT (cit. on p. 55).

[26] Michael Robbins. «Change-Point Analysis: Asymptotic Theory and Applications». All Dissertations. 391. PhD thesis. 2009 (cit. on pp. 60, 61, 93).

[27] M. Csörgö and L. Horváth. *Weighted Approximations in Probability and Statistics*. Wiley Series in Probability and Statistics. Wiley, 1993 (cit. on p. 61).

[28] D.V. Rosato, D.V. Rosato, and M.G. Rosato. *Injection Molding Handbook*. Kluwer Academic Publishers, 2000. ISBN: 9780792386193. URL: https://books.google.it/books?id=l5jqDRauKNYC (cit. on p. 70).

[29] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. «Equation of State Calculations by Fast Computing Machines». In: *The Journal of Chemical Physics* 21.6 (1953), pp. 1087–1092. DOI: 10.1063/1.1699114 (cit. on p. 70).

[30] A. Gelman, W. R. Gilks, and G. O. Roberts. «Weak convergence and optimal scaling of random walk Metropolis algorithms». In: *The Annals of Applied Probability* 7.1 (1997), pp. 110–120. DOI: 10.1214/aoap/1034625254 (cit. on p. 72).

[31] D. Berndt and J. Clifford. «Using Dynamic Time Warping to Find Patterns in Time Series». In: *KDD Workshop*. 1994 (cit. on p. 88).

[32] Usue Mori, Alexander Mendiburu, and Jose A. Lozano. «Distance Measures for Time Series in R: The TSdist Package». In: *R journal* 8.2 (2016), pp. 451–459. URL: https://journal.r-project.org/archive/2016/RJ-2016-058/index.html (cit. on p. 88).

[33] Dirk Eddelbuettel and Romain François. «Rcpp: Seamless R and C++ Integration». In: *Journal of Statistical Software* 40.8 (2011), pp. 1–18. DOI: 10.18637/jss.v040.i08 (cit. on p. 91).

[34] Dirk Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. New York: Springer, 2013. DOI: 10.1007/978-1-4614-6868-4 (cit. on p. 91).

[35] Dirk Eddelbuettel and James Joseph Balamuta. «Extending extitR with extitC++: A Brief Introduction to extitRcpp». In: *PeerJ Preprints* 5 (2017), e3188v1. DOI: 10.7287/peerj.preprints.3188v1 (cit. on p. 91).